Max 9 JS API

Version 9.0.6-rev.1

Cycling ’74

cycling74.com

Contents

Max 9 JS API

Reference for the API available within the `[js]`, `[jsui]`, `[v8]`, `[v8.codebox]`, and `[v8ui]` objects

Classes

Item	Description
Buffer	Bind to a Max `buffer~` object.
Dict	Bind to a Max `dict` object.
File	The File object provides a means of reading and writing files from JavaScript.
Folder	Iterate through the files in a folder.
Global	Global object for sharing data between Max JavaScript instances.
Image	Bitmap image object handle
Jitter3dUtilsInterface	Utilities for Jitter 3D manipulations
JitterEvent	The argument passed to a JitterListener callback function.
JitterListener	A listener for changes in a JitterObject.
JitterMatrix	A named matrix which may be used for data storage and retrieval,

	resampling, and matrix type and planecount conversion operations.
JitterObject	A JavaScript representation of a Jitter object in a patcher.
jsthis	The "this" object in the context of JavaScript in Max
LiveAPI	A means of communicating with the Live API from JavaScript.
Max	Singleton Max object, controlling the Max environment.
Maxobj	A JavaScript representation of a Max object in a patcher.
MaxobjConnection	A JavaScript representation of a patchcord connection.
MaxobjListener	A listener for changes in a Maxobj object.
MaxobjListenerData	The argument provided to a MaxobjListener callback function.
MaxString	Bind a Max `string` object
MGraphics	Drawing context for rendering shapes and images.
MGraphicsSVG	SVG object handle
ParameterInfoProvider	Provides a list of named parameter objects within a patcher hierarchy as well as information about specific parameter objects. It can also notify when parameter objects are added or removed from a patcher hierarchy.
ParameterInfoProviderData	The argument to the ParameterInfoProvider's callback function
ParameterListener	A listener for changes in named parameters.
ParameterListenerData	The argument provided to a ParameterListener callback function.
Patcher	A JavaScript representation of a Max patcher.
Pattern	Pattern object for drawing gradients and patterns.
PolyBuffer	Bind to a Max `polybuffer~` object.
Sketch	Interface to an OpenGL-backed drawing context
SnapshotAPI	Provides access to patcher snapshots.
SQLite	Provides access to the SQLite database system.

SQLResult	A container for results obtained in an SQLite.exec() call.
Task	A function that can be scheduled or repeated.
Wind	A property of the Patcher which represents its window.

Enums

Item	Description
Basic2dStrokeStyleParameterNames	Stroke parameters for use with Sketch.beginstroke() in the "basic2d" drawing style
LineStrokeStyleParameterNames	Stroke parameters for use with Sketch.beginstroke() in the "line" drawing style

Functions

Item	Description
cpost	Prints a message to the system console window. See post() for more details about arguments and formatting.
error	Prints a message to the Max console with a red tint. See post() for more details about arguments and formatting.
messnamed	Sends a message to the named Max object.
post	Prints a representation of the arguments in the Max window.

Namespaces

Item	Description
FileTypes	Types used with
Jitter3dUtilsTypes	Types used with Jitter3dUtilsInterface
JitterEventTypes	Possible event types for a JitterEvent

Type Aliases

Item	Description
DrawingPrimitiveType	Primitive type to use for drawing shapes. See Sketch.shapeprim().
MGraphicsMatrixHandle	Opaque handle for an MGraphics transformation matrix.
MGraphicsPathHandle	Opaque handle for an MGraphics path.

enum Basic2dStrokeStyleParameterNames

Stroke parameters for use with Sketch.beginstroke() in the "basic2d" drawing style

Members

Member	Value	Description
alpha	`"alpha"`	May vary point to point. Value is specified as an alpha value. Useful if alpha is the only color channel which will vary throughout the path.
color	`"color"`	May vary point to point. Values are specified as red, green, blue, alpha.
order	`"order"`	Global for a given path. Value must be interpolation order. Default is 3, or bi-cubic interpolation.
outcolor	`"outcolor"`	May vary point to point. Outline color. Values are specified as red, green, blue, and alpha values. If no color is specified, then the outline color will be the same as the interior color.
outline	`"outline"`	Global for a given path. Value is 0 (off) or 1 (on). Default is 1.
scale	`"scale"`	May vary point to point. Value is specified as an width value. Width of the stroked path.
slices	`"slices"`	Global for a given path. Number of slices for a curved section. Default is 20.

class Buffer

Bind to a Max `buffer~` object.

The Buffer object in JavaScript is a companion to the buffer~ object in Max. Through it, you can to access samples and metadata for the buffer~ object with the given name.

Constructors

```
new Buffer(name: string);
```

Constructs a new instance of the `Buffer` class

Parameter	Type	Description
name	string	name of the Max `buffer~` to bind to.

Methods

channelcount

Buffer channel count

```
channelcount(): number;
```

Name	Type	Description
Return Value	number	The number of channels in the buffer~ object.

framecount

Buffer frame count

Return the number of frames (samples in a single channel) in the buffer~ object.

```
framecount(): number;
```

Name	Type	Description
Return Value	number	The number of frames in the buffer~ object.

freepeer

Free the native C peer

Frees the buffer~ data from the native C peer (created when making a object), which is not considered by the JavaScript garbage collector, and may consume lots of memory until the garbage collector decides to run based JS allocated memory. Once called, the object is not available for any other use. It's not necessary to call this function, as the memory will be freed eventually, but you can call it whenever you're done with your object.

```
freepeer(): void;
```

length

Length of the buffer in milliseconds

```
length(): number;
```

Name	Type	Description
Return Value	number	

peek

Fetch an array of samples from the buffer

```
peek(channel: number, frame: number, count: number): number[];
```

Name	Type	Description
channel	number	channel to fetch samples from (indexed from 1)

frame	number	frame at which to start fetching samples (indexed from 0)
count	number	number of samples to fetch
Return Value	number[]	

poke

Write samples into the buffer

It is more efficient to call this function once with an array, than to call it mulpitle times, each time with a single sample.

```
poke(channel: number, frame: number, samples: number | number[]): void;
```

Name	Type	Description
channel	number	channel to write samples to (indexed from 1)
frame	number	frame at which to start writing samples (indexed from 0)
samples	number | number[]	samples to write (or a single sample to write)

send

Send a message to the buffer.

Can send any message that buffer~ understands. See buffer~ reference.

```
send(message: string, ...args: any[]): void;
```

Name	Type	Description
message	string	the name of message
args	any[]	arguments that follow the name of the message

function cpost

Prints a message to the system console window. See post() for more details about arguments and formatting.

```
export declare function cpost(...args: any): void;
```

Name	Type	Description
args	any	one or more arguments to print

class Dict

Bind to a Max `dict` object.

The Dict object lets you access structured data (a dictionary) associated with a name. If there is a named `dict` object in Max, it will share its contents with the JavaScript Dict object of the same name.

Constructors

```
new Dict(name?: string);
```

Create a named dictionary

If no name is provided as an argument then a unique name will be generated for the dictionary.

Parameter	Type	Description
optional name	string	name of the dictionary

Properties

name string

The name of the dictionary

quiet boolean

Suppresses many errors or warnings if set to true

Methods

append

Add values to the associated data

Add values to the end of an array associated with the specified key.

```
append(key: string, ...values: any[]): void;
```

Name	Type	Description
key	string	the entry key
values	any[]	

clear

Erase the contents of the dictionary, restoring to a clean state.

```
clear(): void;
```

clone

Copy the named dictionary into this dictionary

```
clone(name: string): void;
```

Name	Type	Description
name	string	name of the dictionary to copy

Example

```
var d1 = new Dict("one");
var d2 = new Dict("two");

d1.clone("two"); // Copies the dictionary named "two" into d1
```

contains

Return a 0 or 1 indicating the specified key exists (or doesn't) in the dictionary.

```
contains(key: string): number;
```

Name	Type	Description
key	string	name of the key to lookup
Return Value	number	

export_json

Write a file in the JSON format.

```
export_json(filename: string): void;
```

Name	Type	Description
filename	string	Name of the file to read

export_yaml

Write a file in the YAML format.

```
export_yaml(filename: string): void;
```

Name	Type	Description
filename	string	Name of the file to read

freepeer

Free the native C peer

Frees the dictionary data from the native C peer (created when making a Dict object), which is not considered by the JavaScript garbage collector, and may consume lots of memory until the garbage collector decides to run based JS allocated memory. Once called, the Dict object is not available for any other use. It's not necessary to call this function, as the memory will be freed eventually, but you can call it whenever you're done with your Dict object.

```
freepeer(): void;
```

get

Return the value associated with a key.

```
get(key: string): any;
```

Name	Type	Description
key	string	lookup key
Return Value	any	

getkeys

Return a list of all the keys in a dictionary.

```
getkeys(): string[];
```

Name	Type	Description
Return Value	string[]	

getnames

List of all named dictionaries.

Return a list of all the dictionary names that are known to Max.

```
getnames(): string[];
```

Name	Type	Description
Return Value	string[]	

getsize

Length of the array of values associated with a key.

```
getsize(key: string): number;
```

Name	Type	Description
key	string	The key to look up
Return Value	number	

gettype

Type of the value or values associated with a key.

```
gettype(key: string): string;
```

Name	Type	Description
key	string	The key to look up
Return Value	string	

import_json

Read a file in the JSON format.

```
import_json(filename: string): void;
```

Name	Type	Description
filename	string	Name of the file to read

import_yaml

Read a file in the YAML format.

```
import_yaml(filename: string): void;
```

Name	Type	Description
filename	string	Name of the file to read

parse

Parse a serialized dictionary and replace contents

Replaces the contents of the dictionary by parsing a dictionary serialization. Understands JSON and Max Dictionary Syntax

```
parse(serialization: string): void;
```

Name	Type	Description
serialization	string	serialized dictionary string

pull_from_coll

Pull data from a named `coll` object

Adds entries to the dictionary by pulling rows from the given `coll` object. Does not clear existing keys from the dictionary.

```
pull_from_coll(coll_name: string): void;
```

Name	Type	Description
coll_name	string	name of the `coll` object in Max

push_to_coll

Pull data into a named `coll` object

Push the dictionary's content into a named `coll` object. The keys in the dictionary will become the indices in the `coll`, and the values for those indices the values of the dictionary's keys.

```
push_to_coll(coll_name: string): void;
```

Name	Type	Description
coll_name	string	name of the `coll` object in Max

readany

Loads the contents of a file.

Replaced the contents of the dictionary, clearing existing keys. Accepts JSON and YAML files.

```
readany(filename: string): void;
```

Name	Type	Description
filename	string	name of the file to load

remove

Remove a key and its associated value from the dictionary.

```
remove(key: string): void;
```

Name	Type	Description
key	string	the key to remove

replace

Set the value for a key to a specified value, creating a nested dicts if necessary.

Unlike Dict.set(), this function will create a hierarchical path to a given value if one does not already exist.

```
replace(key: string, ...value: any[]): void;
```

Name	Type	Description
key	string	The key or path to a dictionary entry
value	any[]	The value or values to store at that path

Example

```
var d = new Dict(); // empty dictionary
d.replace("first::second::third", 42);
post(d.get("first").get("second").get("third")); // 42
```

set

Set the value for a key to a specified value.

Unlike Dict.replace(), will not create nested dictionaryies if the nested structure does not already exist.

```
set(key: string, ...value: any[]): void;
```

Name	Type	Description
key	string	The key or path to a dictionary entry
value	any[]	The value or values to store at that path

setparse

Set the value for a key using a serialized dictionary

The serialization can be formatted as JSON or as Max Dictionary Syntax

```
setparse(key: string, serialization: string): void;
```

Name	Type	Description
key	string	The key or path to a dictionary entry
serialization	string	Will be parsed to a dictionary and added to the Dict

stringify

Return the content of the dictionary as a JSON string.

```
stringify(): string;
```

Name	Type	Description
Return Value	string	

writeagain

Open a save dialog to write the dictionary contents to a file.

```
writeagain(): void;
```

type DrawingPrimitiveType

Primitive type to use for drawing shapes. See Sketch.shapeprim().

```
export declare type DrawingPrimitiveType =
        "lines"
        | "line_loop"
        | "line_strip"
        | "points"
        | "polygon"
        | "quads"
        | "quad_grid"
        | "quad_strip"
        | "triangles"
        | "tri_grid"
        | "tri_fan"
        | "tri_strip";
```

function error

Prints a message to the Max console with a red tint. See post() for more details about arguments and formatting.

```
export declare function error(...args: any): void;
```

Name	Type	Description
args	any	one or more arguments to print

class File

The File object provides a means of reading and writing files from JavaScript.

Constructors

```
new File(filename?: string, access?: FileTypes.FileAccess, typelist?:
FileTypes.FourCharacterCode[]);
```

Create a file reference for reading or writing

By default, `typelist` is empty. If `filename` includes an extension, then it is not necessary to supply a `typelist`. If `filename` does not include an extension, then will look for a file with one of the extensions specified by `typelist`. Note that the four-character code may not match the file extension.

Given arguments, the `File` constructor will open the file automatically if the file can be opened.

Parameter	Type	Description
optional filename	string	File to open. Can be relative, absolute, or anything in the Max search path
optional access	FileTypes.FileAccess	Access mode. Can be "read", "write", or "readwrite"
optional typelist	FileTypes.FourCharacterCode[]	Any of the four character codes in max-fileformats.txt

Example 1

```
var f = new File("myfile.txt", "write");
post(f.isopen); // true, if myfile.txt is in the Max search path
f.close();
```

It's also possible to create an empty `File` object, and to configure it after the fact. In this case, the `File` object must be opened explicitly.

Example 2

```
var f = new File();
f.filename = "myfile.txt";
f.access = "write";
f.open();
post(f.isopen); // true, if myfile.txt is in the Max search path
f.close();
```

If for some reason the file could not be opened, then will return false.

Properties

access FileTypes.FileAccess

File access permissions: "read", "write", or "readwrite". By default, this value is "read".

byteorder FileTypes.FileEndianness

The assumed file byteorder (endianness): "big", "little", or "native". By default, this value is "native".

eof number

The location of the end of file, in bytes. If set past the end of current file, `File` will append NULL bytes.

filename string

The current filename.

filetype FileTypes.FourCharacterCode

The four-character code for the file type.

foldername string

read-only

The absolute path to parent folder.

isopen boolean

read-only

Return a true/false indicating if the File constructor is successful in finding and opening the file.

linebreak FileTypes.FileLineEndingStyle

The line break convention to use when writing lines. Defaults to "native"

position number

The current file position, in bytes. Set this to offset the file write position forward or backwards.

typelist FileTypes.FourCharacterCode[]

An array file type codes to filter by when opening a file. By default, this is the empty array.

Methods

close

Closes the currently open file.

```
close(text?: string): void;
```

Name	Type	Description
optional text	string	unclear?

open

Opens the file specified by the filename argument. If no argument is specified, it will open the last opened file, or the value stored in the `filename` property. Check to see if the file was openend successfully.

```
open(filename?: string): void;
```

Name	Type	Description
optional filename	string	the file to open

readbytes

Read bytes to an array

Reads and returns an array containing up to `count` numbers, read as bytes from the file, starting at the current file position. The file position is updated accordingly.

```
readbytes(count: number): number[];
```

Name	Type	Description
count	number	maximum number of bytes to read
Return Value	number[]	

readchars

Read bytes to an array of single character strings

Reads and returns an array containing the single character strings, read as characters from the file, starting at the current file position. The file position is updated accordingly.

```
readchars(count: number): string[];
```

Name	Type	Description
count	number	maximum number of bytes to read
Return Value	string[]	

readfloat32

Read from a file as 32-bit floating point numbers

Reads and returns an array containing the numbers read as 32-bit floating point numbers from the file, starting at the current file position. The byteorder property is taken into account when reading these values. The file position is updated accordingly.

```
readfloat32(count: number): number[];
```

Name	Type	Description
count	number	maximum number of floats to read
Return Value	number[]	

readfloat64

Read from a file as 64-bit floating point numbers

Reads and returns an array containing the numbers read as 64-bit floating point numbers from the file, starting at the current file position. The byteorder property is taken into account when reading these values. The file position is updated accordingly.

```
readfloat64(count: number): number[];
```

Name	Type	Description
count	number	maximum number of floats to read
Return Value	number[]	

readint16

Read from a file as 16-bit signed integers

Reads and returns an array containing the numbers read as signed 16-bit integers from the file, starting at the current file position. The byteorder property is taken into account when reading

these values. The file position is updated accordingly.

```
readint16(count: number): number[];
```

Name	Type	Description
count	number	maximum number of integers to read
Return Value	number[]	

readint32

Read from a file as 32-bit signed integers

Reads and returns an array containing the numbers read as signed 32-bit integers from the file, starting at the current file position. The byteorder property is taken into account when reading these values. The file position is updated accordingly.

```
readint32(count: number): number[];
```

Name	Type	Description
count	number	maximum number of integers to read
Return Value	number[]	

readline

Read a line of text

Reads and returns a string containing up to `maximumCount` characters or up to the first line break as read from the file, starting at the current file position. The file position is updated accordingly The default maximum count value is 512. This can be increased by specifying a new maximum count as the argument.

```
readline(maximumCount: number): string;
```

Name	Type	Description
maximumCount	number	maximum string length to read
Return Value	string	

readstring

Read a string of text

Reads and returns a string containing up to char_count characters as read from the file, starting at the current file position. Unlike , line breaks are not considered. The file position is updated accordingly.

```
readstring(count: number): string;
```

Name	Type	Description
count	number	maximum number of bytes to read
Return Value	string	

writebytes

Write an array of bytes

Writes the numbers contained in the `bytes` argument as bytes to the file, starting at the current file position. The file position is updated accordingly.

```
writebytes(bytes: number[]): void;
```

Name	Type	Description
bytes	number[]	bytes to write

writechars

Write single character strings

Writes the single character strings contained in the `chars` argument as characters to the file, starting at the current file position. The file position is updated accordingly.

```
writechars(chars: string[]): void;
```

Name	Type	Description
chars	string[]	array of single character strings

writefloat32

Write numbers as 32-bit floating point numbers

Writes the numbers contained in the `floats` argument as 32-bit floating point numbers to the file, starting at the current file position. The byteorder property is taken into account when writing these values. The file position is updated accordingly.

```
writefloat32(floats: number[]): void;
```

Name	Type	Description
floats	number[]	array of numbers to write as float32

writefloat64

Write numbers as 64-bit floating point numbers

Writes the numbers contained in the `floats` argument as 64-bit floating point numbers to the file, starting at the current file position. The byteorder property is taken into account when writing these values. The file position is updated accordingly.

```
writefloat64(floats: number[]): void;
```

Name	Type	Description
floats	number[]	array of numbers to write as float64

writeint16

Write numbers as 16-bit signed integers

Writes the numbers contained in the `ints` argument as signed 16-bit integers to the file, starting at the current file position. The byteorder property is taken into account when writing these values. The file position is updated accordingly.

```
writeint16(ints: number[]): void;
```

Name	Type	Description
ints	number[]	array of numbers to write as int16

writeint32

Write numbers as 32-bit signed integers

Writes the numbers contained in the `ints` argument as signed 32-bit integers to the file, starting at the current file position. The byteorder property is taken into account when writing these values. The file position is updated accordingly.

```
writeint32(ints: number[]): void;
```

Name	Type	Description
ints	number[]	array of numbers to write as int32

writeline

Write a line of text

Writes the characters contained in the string argument as characters to the file, starting at the current file position, and inserts a line break appropriate to the `linebreak` property. The file position is updated accordingly.

```
writeline(text: string): void;
```

Name	Type	Description
text	string	the text to write

writestring

Write a string of text

Writes the characters contained in the string argument as characters to the file, starting at the current file position. Unlike , no line break is inserted. The file position is updated accordingly.

```
writestring(text: string): void;
```

Name	Type	Description
text	string	the text to write

namespace FileTypes

Types used with

type FileAccess

access type

```
export type FileAccess = "read" | "write" | "readwrite";
```

type FileEndianness

endianness

```
export type FileEndianness = "big" | "little" | "native";
```

type FileLineEndingStyle

line ending style

```
export type FileLineEndingStyle = "dos" | "mac" | "unix" | "native";
```

type FourCharacterCode

Four character file codes.

```
export type FourCharacterCode = "iLaF" | "maxb" | "TEXT" | "mx@c" | "GenX"
| "mRef" | "cafe" | "jar " | "WAVE" | "wv64" | "AIFF" | "NxTS" | "ULAW" |
"FLAC" | "DATA" | "Midi" | "JiT!" | "maxc" | "PICT" | "PICS" | "MPEG" |
"mpg4" | "MooV" | "WVC1" | "WMVA" | "WMV3" | "WMV2" | "M4V " | "GIFf" |
"JPEG" | "PNG " | "PNGf" | "TIFF" | "SWFL" | "8BPS" | "BMP " | "exr " |
"VfW " | "AFxP" | "AFxB" | "AUps" | "V3ps" | "JSON" | "mSnp" | "mPrj" |
"mZip" | "mPak" | "zip " | "DICT" | "YAML" | "svg " | "css " | "XSLT" |
"ampf" | "amxd" | "pStx" | "pSto" | "gDSP" | "gJIT" | "Jmtl" | "Jobj" |
"Jdae" | "Jbln" | "J3ds" | "Jase" | "Jply" | "Jdxf" | "Jlwo" | "Jlxo" |
"Jstl" | "Jac " | "Jmsd" | "Jcob" | "Jscb" | "Jsmd" | "Jvta" | "Jmdl" |
"Jmd2" | "Jmd3" | "Jpk3" | "Jmdc" | "Jmd5" | "Jbvh" | "Jcsm" | "Jxmd" |
"Jb3d" | "Jq3d" | "Jq3s" | "Jogr" | "Jirm" | "Jirr" | "Jnff" | "Js8w" |
"Joff" | "Jraw" | "Jter" | "J3dm" | "Jhmp" | "Jndo" | "FBX " | "glTF" |
"Mp3 " | "M4a " | "CAF " | "OGG " | "mQur" | "mLsn" | "Jlua" | "mMap" |
"mxPL" | "mxCT" | "mUgh" | "mMtr" | "mTXT" | "RBOP" | "aPin" | "aPcs" |
"a3in" | "a3cs" | "AUpi" | "AUin" | "APPL" | "xQZZ" | "TXT ";
```

class Folder

Iterate through the files in a folder.

Example

```
f = new Folder("patches");
// would try to find the patches folder in the search path
f = new Folder("Disk:/folder1/folder2");
// uses an absolute path
```

After creating a Folder object, you'll probably want to restrict the files you see while traversing it by setting the typelist property:

```
f.typelist = ["iLaF", "maxb", "TEXT"];
// typical max files
```

Check the file max-fileformats.txt inside the init folder in the Cycling ’74 folder for filetype codes and their associated extensions. As a Folder object traverses through the files, you can find out information about the current file using its file properties. You can also determine whether you've looked at all properties by testing the end property. The following code prints the names of all files found in the folder.

```
while (!f.end) {
 post(f.filename);
 post();
 f.next();
}
```

To finish with the Folder object, you can either delete it, or send it the close message if you might want to reuse it.

```
f.close();
```

Two types of properties of a Folder are available: some refer to the current file within the folder, and some refer to the Folder object’s state. Most of these properties are read-only.

Constructors

```
new Folder(pathname: string);
```

Constructs a new instance of the `Folder` class

pathname can be in the search path or a complete pathname using Max path syntax.

Parameter	Type	Description
pathname	string	the name of a folder

Properties

count number

read-only

The total number of files of the specified type(s) contained in the folder.

end boolean

read-only

Non-zero (true) if there are no more files to examine in the folder, or if the pathname argument to the Folder object didn't find a folder.

extension string | null

read-only

The extension of the current file's name, including the period. If there are no characters after the period, a null value is returned.

filename string

read-only

The name of the current file.

filetype string | null

read-only

The four-character code associated with the current file's filetype. These codes are listed in the file max-fileformats.txt, which is located at /Library/Application Support/Cycling ’74 on Macintosh and C:\Program Files\Common Files\Cycling ’74 on Windows. If there is no mapping for the file's extension, a null value is returned.

moddate any[]

read-only

An array containing the values year, month, day, hour, minute, and second with the last modified date of the current file. These values can be used to create a Javascript Date object.

pathname string

read-only

The full pathname of the folder

typelist string[]

The list of file types that will be used to find files in the folder. To search for all files (the default), set the typelist property to an empty array.

Methods

close

Closes the folder. To start using it again, call the reset() function.

```
close(): void;
```

next

Moves to the next file.

```
next(): void;
```

reset

Start iterating at the beginning of the list of files. Re-opens the folder if it was previously closed with the close() function.

```
reset(): void;
```

class Global

Global object for sharing data between Max JavaScript instances.

Each Global object is a reference to a shared storage object. Creating two Global objects with the same namespace will share stored properties. Properties stored in a Global object can be accessed in Max, from outside of JavaScript.

Example 1

```
var g = new Global("name");
g.bob = 12;
var h = new Global("name");
post(h.bob); // 12
```

You can send messages to a named Global object from Max, using a message object.

Example 2

```
; xyz height 42
// Put this in a message box to set the property `height` of the Global
object
// named `xyz` to the value 42.

; xyz frank x y z
// Set the value of `frank` of the Global object `xyz` to the string array
["x", "y", "z"].

; xyz sendnamed height dest
// Send the value of `height` of the Global object `xyz` to all receive
objects named `dest`.
```

Constructors

```
new Global(namespace: string);
```

Constructs a new instance of the `Global` class

Parameter	Type	Description
namespace	string	Namespace identifier

Methods

sendnamed

Forward a property to receive objects

Forward the value of a property to named `receive` objects. The `target` of `sendnamed` is the name of a `receive` object, in which case every `receive` object with that name will receive the value of the named property.

```
sendnamed(target: string, propertyName: string): void;
```

Name	Type	Description
target	string	Name of a `receive` object
propertyName	string	Identifier for a property stored in the Global object

Example

```
var g = new Global("xyz");
g.ethyl = 1000;
g.sendnamed("fred", "ethyl");
// every [receive fred] will output 1000
```

class Image

Bitmap image object handle

You can use MGraphics.image_surface_draw() and MGraphics.set_source_surface() to draw using a bitmap Image. Create an Image either using a file in Max's search path, or from an existing MGraphics context.

Example

```
function paint() {

                // Render a simple image
                var im = new Image("icon.png");
                mgraphics.image_surface_draw(im);

                // Render from an offscreen mgraphics
                var offscreen_ctx = new MGraphics(200, 200);
                offscreen_ctx.rectangle(10, 10, 50, 50);
                offscreen_ctx.set_source_rgba(1, 0, 0, 1);
                offscreen_ctx.fill();
                var im2 = new Image(offscreen_ctx);

                mgraphics.image_surface_draw(im2);
}
```

Constructors

```
new Image(source: MGraphics | string);
```

Create a new bitmap image

Parameter	Type	Description
source	MGraphics | string	Filename in Max's search path, or an MGraphics context

Properties

size [number, number]

read-only

Get a the width and height of the image

Methods

flip

Flip the image

```
flip(horizontal: 0 | 1, vertical: 0 | 1);
```

Name	Type	Description
horizontal	0 | 1	
vertical	0 | 1	

class Jitter3dUtilsInterface

Utilities for Jitter 3D manipulations

It is not necessary to instantiate this interface directly; it is available globally as a `Jitter3DUtils` object and methods can be called like `Jitter3DUtils.vadd()`.

Methods

add_quats

Add three 4D vectors (quaternions)

```
add_quats(q1: Jitter3dUtilsTypes.vec4, q2: Jitter3dUtilsTypes.vec4, q3:
Jitter3dUtilsTypes.vec4): void;
```

Name	Type	Description
q1	Jitter3dUtilsTypes.vec4	first quaternion
q2	Jitter3dUtilsTypes.vec4	second quaternion
q3	Jitter3dUtilsTypes.vec4	third quaternion

axis_to_quat

Convert an angle/axis rotation to a quaternion

```
axis_to_quat(axis: Jitter3dUtilsTypes.vec3, quat:
Jitter3dUtilsTypes.vec4): void;
```

Name	Type	Description
axis	Jitter3dUtilsTypes.vec3	axis rotation
quat	Jitter3dUtilsTypes.vec4	output quaternion

build_rotmatrix

Build a rotation matrix for a given quaternion

```
build_rotmatrix(m: Jitter3dUtilsTypes.vec4, q: Jitter3dUtilsTypes.vec4):
void;
```

Name	Type	Description
m	Jitter3dUtilsTypes.vec4	rotation matrix
q	Jitter3dUtilsTypes.vec4	quaternion

closest_line_sphere

Set `p1` to the point on a sphere closest to a line segment

```
(x3 - x1)(x2 - x1) + (y3 - y1)(y2 - y1) + (z3 - z1)(z2 - z1)
-----------------------------------------------------------
(x2 - x1)(x2 - x1) + (y2 - y1)(y2 - y1) + (z2 - z1)(z2 - z1)
```

```
closest_line_sphere(lineA: Jitter3dUtilsTypes.vec3, lineB:
Jitter3dUtilsTypes.vec3, center: Jitter3dUtilsTypes.vec3, r: number, p1:
Jitter3dUtilsTypes.vec3): void;
```

Name	Type	Description
lineA	Jitter3dUtilsTypes.vec3	first line point
lineB	Jitter3dUtilsTypes.vec3	second line point
center	Jitter3dUtilsTypes.vec3	center point
r	number	sphere radius
p1	Jitter3dUtilsTypes.vec3	point to set

intersect_line_quad

Returns whether the ray defined by the line's two points intersect the quad defined by a position, rotation, and scale. Sets `p2` to the point of intersection with the quad plane in unit coordinates and sets `p1` to the same point in world coordinates.

```
intersect_line_quad(lineA: Jitter3dUtilsTypes.vec3, lineB:
Jitter3dUtilsTypes.vec3, pos: Jitter3dUtilsTypes.vec3, rot:
Jitter3dUtilsTypes.vec3, scale: Jitter3dUtilsTypes.vec3, p1:
Jitter3dUtilsTypes.vec3, p2: Jitter3dUtilsTypes.vec3): boolean;
```

Name	Type	Description
lineA	Jitter3dUtilsTypes.vec3	first line point
lineB	Jitter3dUtilsTypes.vec3	second line point
pos	Jitter3dUtilsTypes.vec3	quad position
rot	Jitter3dUtilsTypes.vec3	quad rotation
scale	Jitter3dUtilsTypes.vec3	quad scale
p1	Jitter3dUtilsTypes.vec3	world coordinate intersection point
p2	Jitter3dUtilsTypes.vec3	unit coordinate intersection point

intersect_line_sphere

Returns whether the ray defined by the line's two points intersect with the sphere of given center and radius. Sets `p1` to the closest point of intersection.

```
intersect_line_sphere(lineA: Jitter3dUtilsTypes.vec3, lineB:
Jitter3dUtilsTypes.vec3, center: Jitter3dUtilsTypes.vec3, r: number, p1:
Jitter3dUtilsTypes.vec3): boolean;
```

Name	Type	Description
lineA	Jitter3dUtilsTypes.vec3	first line point
lineB	Jitter3dUtilsTypes.vec3	second line point
center	Jitter3dUtilsTypes.vec3	sphere center
r	number	sphere radius
p1	Jitter3dUtilsTypes.vec3	closest point of intersection
Return Value	boolean	

normalize_quat

Normalize a quaternion

```
normalize_quat(quat: Jitter3dUtilsTypes.vec4): void;
```

Name	Type	Description
quat	Jitter3dUtilsTypes.vec4	a quaternion to normalize

quat_to_axis

Convert a quaternion to an angle/axis rotation

```
quat_to_axis(quat: Jitter3dUtilsTypes.vec4, axis:
Jitter3dUtilsTypes.vec3): void;
```

Name	Type	Description
quat	Jitter3dUtilsTypes.vec4	quaternion to convert
axis	Jitter3dUtilsTypes.vec3	output angle/axis rotation

transform_point

```
transform_point(point: Jitter3dUtilsTypes.vec4, matrix:
Jitter3dUtilsTypes.mat16): void;
```

Name	Type	Description
point	Jitter3dUtilsTypes.vec4	a point to transform
matrix	Jitter3dUtilsTypes.mat16	a 4x4 matrix

vadd

Add two 3D vectors

```
vadd(src1: Jitter3dUtilsTypes.vec3, src2: Jitter3dUtilsTypes.vec3, dst:
Jitter3dUtilsTypes.vec3): void;
```

Name	Type	Description
src1	Jitter3dUtilsTypes.vec3	first vector
src2	Jitter3dUtilsTypes.vec3	second vector
dst	Jitter3dUtilsTypes.vec3	output vector to store sum of `src1 + src2`

vcopy

Copy one vector to another

```
vcopy(v1: Jitter3dUtilsTypes.vec3, v2: Jitter3dUtilsTypes.vec3): void;
```

Name	Type	Description
v1	Jitter3dUtilsTypes.vec3	source
v2	Jitter3dUtilsTypes.vec3	destination

vcross

Calculate the cross product of two 3D vectors

```
vcross(v1: Jitter3dUtilsTypes.vec3, v2: Jitter3dUtilsTypes.vec3, cross:
Jitter3dUtilsTypes.vec3): void;
```

Name	Type	Description
v1	Jitter3dUtilsTypes.vec3	first vector
v2	Jitter3dUtilsTypes.vec3	second vector
cross	Jitter3dUtilsTypes.vec3	cross product

vdiv

Divide `src1` and `src2` (element-wise) and store the result in `dst`

```
vdiv(src1: Jitter3dUtilsTypes.vec3, src2: Jitter3dUtilsTypes.vec3, dst:
Jitter3dUtilsTypes.vec3): void;
```

Name	Type	Description
src1	Jitter3dUtilsTypes.vec3	first vector
src2	Jitter3dUtilsTypes.vec3	second vector
dst	Jitter3dUtilsTypes.vec3	destination vector

vdot

Calculate the dot product of two 3D vectors

```
vdot(v1: Jitter3dUtilsTypes.vec3, v2: Jitter3dUtilsTypes.vec3): number;
```

Name	Type	Description
v1	Jitter3dUtilsTypes.vec3	first vector
v2	Jitter3dUtilsTypes.vec3	second vector
Return Value	number	

vlength

Compute the squared distance of a 3D vector

```
vlength(v: Jitter3dUtilsTypes.vec3): number;
```

Name	Type	Description
v	Jitter3dUtilsTypes.vec3	vector
Return Value	number	

vlength2

A cheaper distance-squared calculation

```
vlength2(v: Jitter3dUtilsTypes.vec3): number;
```

Name	Type	Description
v	Jitter3dUtilsTypes.vec3	vector
Return Value	number	

vmul

Multiply `src1` and `src2` (element-wise) and store the result in `dst`

```
vmul(src1: Jitter3dUtilsTypes.vec3, src2: Jitter3dUtilsTypes.vec3, dst:
Jitter3dUtilsTypes.vec3): void;
```

Name	Type	Description
src1	Jitter3dUtilsTypes.vec3	first vector
src2	Jitter3dUtilsTypes.vec3	second vector
dst	Jitter3dUtilsTypes.vec3	destination vector

vnormal

Normalize a 3D vector

```
vnormal(v: Jitter3dUtilsTypes.vec3): void;
```

Name	Type	Description
v	Jitter3dUtilsTypes.vec3	vector

vscale

Scale a vector

```
vscale(v: Jitter3dUtilsTypes.vec3, scale: number): void;
```

Name	Type	Description
v	Jitter3dUtilsTypes.vec3	vector
scale	number	scale factor

vset

Set the values of a vector

```
vset(v: Jitter3dUtilsTypes.vec3, x: number, y: number, z: number): void;
```

Name	Type	Description
v	Jitter3dUtilsTypes.vec3	vector
x	number	new x value
y	number	new y value
z	number	new z value

vsub

Subtract `src2` from `src1` and store the result in `dst`

```
vsub(src1: Jitter3dUtilsTypes.vec3, src2: Jitter3dUtilsTypes.vec3, dst:
Jitter3dUtilsTypes.vec3): void;
```

Name	Type	Description
src1	Jitter3dUtilsTypes.vec3	first vector
src2	Jitter3dUtilsTypes.vec3	second vector
dst	Jitter3dUtilsTypes.vec3	destination vector

vzero

Set all elements of a vector to zero

```
vzero(v: Jitter3dUtilsTypes.vec3): void;
```

Name	Type	Description
v	Jitter3dUtilsTypes.vec3	vector

xyz_to_axis

Convert rotation in Euler angles (xyz) to angle/axis rotation

```
xyz_to_axis(v: Jitter3dUtilsTypes.vec3, axis: Jitter3dUtilsTypes.vec3):
void;
```

Name	Type	Description
v	Jitter3dUtilsTypes.vec3	rotation in Euler angles
axis	Jitter3dUtilsTypes.vec3	output angle/axis rotation

namespace Jitter3dUtilsTypes

Types used with Jitter3dUtilsInterface

type mat16

A 4x4 matrix

```
export type mat16 = [number, number, number, number, number, number,
number, number,
                number, number, number, number, number, number, number,
number];
```

type vec3

A 3D vector

```
export type vec3 = [number, number, number];
```

type vec4

A 4D vector

```
export type vec4 = [number, number, number, number];
```

class JitterEvent

The argument passed to a JitterListener callback function.

Properties

args any

read-only

Arguments depend on event type

eventname JitterEventTypes.event

read-only

Name of the event to be handled

subjectname string

read-only

Name of the object to listen to

namespace JitterEventTypes

Possible event types for a JitterEvent

type event

Event types

```
export type event = "mouse" | "mouseidle" | "mouseidleout" | "mousewheel"
| "matrix_received" |
                        "messaged_received" | "connected_notification" |
"import" | "collisions" |
                        "matrixoutput" | "swap";
```

class JitterListener

A listener for changes in a JitterObject.

Example

```
var recv = new JitterObject("jit.net.recv");
var mylistener = new JitterListener(recv.getregisteredname(), callbackfun);

function callbackfun(event) {
    if (event.eventname == "matrix_received") {
        matrixoutput(event.args[0]);
    } else if (event.eventname == "message_received") {
        messageoutput(event.args[0]);
    } else if (event.eventname == "connected_notification") {
        connectedoutput();
    }
}
callbackfun.local = 1;
```

Constructors

```
new JitterListener(objectName: string, callback: Function);
```

Constructs a new instance of the `JitterListener` class

Parameter	Type	Description
objectName	string	name of the object to listen to
callback	Function	a function called when a change occurs to the listened-to object which takes a JitterEvent

Properties

function Function

read-only

The callback function to handle the JitterEvent

object JitterObject

read-only

The object being listened to

subjectname string

read-only

Name of the object being listened to

class JitterMatrix

A named matrix which may be used for data storage and retrieval, resampling, and matrix type and planecount conversion operations.

Constructors

```
new JitterMatrix(planeCount?: number, dataType?: string, columns?: number,
rows?: number);
```

Constructs a new instance of the `JitterMatrix` class

Parameter	Type	Description
optional planeCount	number	
optional dataType	string	
optional columns	number	
optional rows	number	

Example

```
var mymatrix = new JitterMatrix(4, "char", 320, 240);
```

Properties

adapt number

Matrix adaptation flag.

When this flag is set (1), the JitterMatrix will adapt to the incoming matrix planecount, type, and dimensions

dim [number, number]

Matrix data dimensions

dimstride [number, number]

Byte stride per dimension

dstdimend number[]

Destination dimension end position (default = all dim values minus 1).

dstdimstart number[]

Destination dimension start position (default = all 0).

interp number

Matrix interpolation flag (default = 0). When the flag is set, the input matrix will be interpolated when copied to the internal matrix.

name string

Name of the matrix (default = UID).

planecount number

Number of planes in the matrix data (default = 4).

planemap number[]

Maps input planes to output planes (default = 0 1 2 3 ...)

size number

Total byte size of matrix.

srcdimend number[]

Source dimension end position (default = all dim values minus 1).

srcdimstart number[]

Source dimension start position (default = all 0).

type string

The matrix data type (default = "char").

- "char": Char data (0-255) - "long": Long data - "float32": 32-bit floating point data - "float64": 64-bit floating point data

usedstdim number

destdim use flag (default = 0).

When the flag is set, the destination dimension's attributes are used when copying an input matrix to an internal matrix.

usesrcdim number

srcdim use flag (default = 0).

When the flag is set, the source dimension's attributes are used used when copying an input matrix to an internal matrix.

Methods

bang

Outputs the currently stored matrix

```
bang(): void;
```

clear

Sets all matrix values to zero

```
clear(): void;
```

copyarraytomatrix

Copy a tightly packed TypedArray to a JitterMatrix

This method is only available in the new v8 javascript engine objects. TypedArrays must match matrix type (Uint8 for char, Int32 for long, Float32 for float32 and Float64 for float64), and their size must match the tighly packed size of the matrix -- i.e. planecount * dim[0] * dim[1] ... * dim[n].

```
copyarraytomatrix(array: Uint8Array | Uint8ClampedArray | Int32Array |
Float32Array | Float64Array): void;
```

Name	Type	Description
array	Uint8Array | Uint8ClampedArray | Int32Array | Float32Array | Float64Array	tightly packed TypedArray

copymatrixtoarray

Copy a JitterMatrix to a tightly packed TypedArray

This method is only available in the new v8 javascript engine objects. TypedArrays must match matrix type (Uint8 for char, Int32 for long, Float32 for float32 and Float64 for float64), and their size must match the tighly packed size of the matrix -- i.e. planecount * dim[0] * dim[1] ... * dim[n].

```
copymatrixtoarray(array: Uint8Array | Uint8ClampedArray | Int32Array |
Float32Array | Float64Array): void;
```

Name	Type	Description
array	Uint8Array | Uint8ClampedArray | Int32Array | Float32Array | Float64Array	tightly packed TypedArray

exportimage

Export the current frame as an image file with the name specified

```
exportimage(filename: string, filetype?: string, useDialog?: number): void;
```

Name	Type	Description
filename	string	exported image filename
optional filetype	string	exported image file type; can be "png", "bmp", "jpeg", "macpaint", "photoshop", "pict", "qtimage", "sgi", "tga", and "tiff" (default = "png")
optional useDialog	number	a value of `1` will open a file dialog to enter image file settings

exportmovie

Export a matrix as a QuickTime movie

```
exportmovie(filename?: string, fps?: number, codec?: string, quality?:
string, timescale?: number): void;
```

Name	Type	Description
optional filename	string	exported movie filename (default = file dialog will open)

optional fps	number	frames per second (default = 30)
optional codec	string	one of "raw", "cinepak", "graphics", "animation", "video", "componentvideo", "jpeg", "mjpegb", "sgi", "planarrgb", "macpaint", "gif", "photocd", "qdgx", "avrjpeg", "opendmljpeg", "bmp", "winraw", "vector", "qd", "h261", "h263", "dvntsc", "dvpal", "dvprontsc", "dvpropal", "flc", "targa", "png", "tiff", "componentvideosigned", "componentvideounsigned", "cmyk", "microsoft", "sorenson", "indeo4", "argb64", "rgb48", "alphagrey32", "grey16", "mpegyuv420", "yuv420", and "sorensonyuv9" (default = "raw")
optional quality	string	one of "lossless", "max", "min", "low", "normal", and "high" (default = "max"). Note that the minimum quality is, in many cases, the codec's default quality. Use "low" quality for consistent results.
optional timescale	number	units per second (default = 600)

exprfill

Evaluate an expression to fill the matrix

If a plane argument is provided, the expression is applied to a single plane. Otherwise, it is applied to all planes in the matrix. See jit.expr for more information on expressions. Unlike the jit.expr object, there is no support for providing multiple expressions to fill multiple planes at once with different expressions. Call this method multiple times for each plane you wish to fill.

```
exprfill(plane: number, expression?: string): void;
```

Name	Type	Description
plane	number	matrix plane to apply to (default = all planes)
optional expression	string	expression to apply

fillplane

Fill the specified plane with a single value

```
fillplane(plane: number, value: number): void;
```

Name	Type	Description
plane	number	plane index
value	number	value to fill with

float

Set all cells to the value specified by value(s) and output the data.

```
float(values: number | number[]): void;
```

Name	Type	Description
values	number | number[]	matrix value list whose length is equal to the dimcount

getcell

Sends the value(s) in the cell specified by position out the right outlet of the object as a list in the form "cell cell - position0 ... cell - positionN val plane0 - value ... planeN - value"

```
getcell(position: [number, number]): void;
```

Name	Type	Description
position	[number, number]	cell position

importmovie

Import a QuickTime movie into the matrix

```
importmovie(filename?: string, timeoffset?: number): void;
```

Name	Type	Description
optional filename	string	filename of movie to import (default = file dialog)
optional timeoffset	number	import time offset (default = 0)

int

Set all cells to the value specified by value(s) and output the data.

```
int(values: number | number[]): void;
```

Name	Type	Description
values	number | number[]	matrix value list whose length is equal to the dimcount

jit_gl_texture

Copy a texture to the matrix

```
jit_gl_texture(texture_name: string): void;
```

Name	Type	Description
texture_name	string	GL texture name

list

Set all cells to the value specified by value(s) and output the data.

```
list(values: number[]): void;
```

Name	Type	Description
values	number[]	matrix value list whose length is equal to the dimcount

op

Perform jit.op operations on the matrix

```
op(operator: string, args: any): void;
```

Name	Type	Description
operator	string	the jit.op operator
args	any	matrix name or constant args for the operator

Example

```
var a = new JitterMatrix(4, "char", 320, 240);
var b = new JitterMatrix(4, "char", 320, 240);
b.setall(255, 255, 255, 255);
a.op("*", b);
```

read

Read Jitter binary data files (.jxf) into the matrix

```
read(filename?: string): void;
```

Name	Type	Description
optional filename	string	binary data file to read (default = file dialog)

setall

Set all cells to the value(s) specified

```
setall(values: number | number[]): void;
```

Name	Type	Description
values	number | number[]	values to set

setcell

Set the cell specified to a value

Arguments are formatted like the `setcell` message to a jit.matrix in Max.

```
setcell(...args: any[]): void;
```

Name	Type	Description
args	any[]	the position, plane, planeNumber, value/values to set

Example

Set cell (20, 30) to (0, 0, 255, 255):

```
mymatrix.setcell(20, 30, "val", 0, 0, 255, 255);
```

setcell1d

Set a cell in a 1D matrix

```
setcell1d(pos: number, ...values: number[]): void;
```

Name	Type	Description
pos	number	cell position
values	number[]	values to set at given position

Example

Set cell (74) to (100, 100, 100, 0):

```
my1dmatrix.setcell1d(74, 100, 100, 100, 0);
```

setcell2d

Set a cell in a 2D matrix

```
setcell2d(posX: number, posY: number, ...values: number[]): void;
```

Name	Type	Description
posX	number	first dimension of cell's position
posY	number	second dimension of cell's position
values	number[]	values to set at given position

Example

Set cell (30, 20) to (255, 255, 0, 0):

```
my2dmatrix.setcell2d(30, 20, 255, 255, 0, 0);
```

setcell3d

Set a cell in a 3D matrix

```
setcell3d(posX: number, posY: number, posZ: number, ...values: number[]):
void;
```

Name	Type	Description
posX	number	first coordinate of a cell's position
posY	number	second coordinate of a cell's position
posZ	number	third coordinate of a cell's position
values	number[]	values to set at given position

Example

Set cell (30, 20, 40) to (100, 200, 300, 0):

```
my3dmatrix.setcell3d(30, 20, 40, 100, 200, 300, 0);
```

setplane1d

Set a cell for a 1D matrix

```
setplane1d(pos: number, plane: number, value: number): void;
```

Name	Type	Description
pos	number	first coordinate
plane	number	plane number
value	number	value to set

setplane2d

Set a cell for a 2D matrix

```
setplane2d(posX: number, posY: number, plane: number, value: number): void;
```

Name	Type	Description
posX	number	first coordinate
posY	number	second coordinate
plane	number	plane number
value	number	value to set

setplane3d

Set a cell for a 3D matrix

```
setplane3d(posX: number, posY: number, posZ: number, plane: number, value:
number): void;
```

Name	Type	Description
posX	number	first coordinate
posY	number	second coordinate
posZ	number	third coordinate
plane	number	plane number
value	number	value to set

val

Set all cells to the value specified

```
val(value: number): void;
```

Name	Type	Description
value	number	value to set

write

Write matrix set as a Jitter binary data file (.jxf)

```
write(filename?: string): void;
```

Name	Type	Description
optional filename	string	name of file to write to (default = file dialog)

class JitterObject

A JavaScript representation of a Jitter object in a patcher.

Example

```
var win = new JitterObject("jit.window", "my_window");
var rend = new JitterObject("jit.gl.render", win.getregisteredname());
```

Constructors

```
new JitterObject(objectName: string, ...params: any);
```

Constructs a new instance of the `JitterObject` class

Parameter	Type	Description
objectName	string	name of Jitter object
params	any	parameter and attributes

Methods

freepeer

Delete the JitterObject

```
freepeer(): void;
```

getregisteredname

Get the registered name of the JitterObject

```
getregisteredname(): string;
```

Name	Type	Description
Return Value	string	

class jsthis

The "this" object in the context of JavaScript in Max

The jsthis object is the "this" within the context of any function you define that can be invoked from Max, as well as in your global code. When you define functions, they become methods of your extension of jsthis. When you use variables in your global code, they become its properties. The jsthis object has certain built-in properties and methods that facilitate interacting with and controlling the Max environment.

You can't create an Instance of this class. It is only available as a global context object in the Max JavaScript environment.

For most messages, a message received in the inlet of the Max `js` object will invoke a method with the same name defined for the JavaScript jsthis object passing anything after the beginning symbol as arguments to the function. The jsthis.bang() and jsthis.list() methods are two examples.

You can also create your own custom message handlers. For example, within Max, sending the message `foo 1 2 3` to the `js` object invokes the `foo()` method of the jsthis object. In other words, it looks for a function property of jsthis called `foo` and passes `1`, `2`, and `3` as arguments to the function.

```
function foo(a, b, c) {
    post(a, b, c);
}
```

Properties

autowatch boolean

Whether automatic file reloading is on

This is particularly useful during development of your Javascript code if you have several js instances using the same source file and you want them all to update when the source changes. It can also be used to facilitate the use of an external text editor. When the text editor saves the file, the js object will notice and recompile it. By default, the value of autowatch is 0 (off). If you want to turn on autowatch, it is best to do so in your global code.

Example

```
// begin file example.js
autowatch = 1;

function loadbang() {
    // your code here
}

// end file example.js
```

box Maxobj

read-only

The current object's box

Example

```
function bang() {
    var box = this.box();
    post(box.rect.toSource());
}
```

editfontsize number

Size of the font shown in the text editing window

Controls the size of the font shown in the text editing window where you edit a script in points. By assigning the editfontsize property in your global code, you can override the default font size setting for text editing, which is the same size as the text shown in the Max window.

*

Example

```
// begin file example.js
editfontsize = 10;

function loadbang() {
    // your code here
}

// end file example.js
```

inlet number

read-only

The inlet number which received the message triggering the currently executing function

During the execution of a function, the inlet property reports the inlet number that received the message that triggered the function, starting at 0 for the leftmost inlet. This property's value is 0 within global code.

Example

```
inlets = 3;

function msg_int(val) {
    switch(inlet) {
        case 2:
            post("inlet 3: " + val);
            break;
        case 1:
            post("inlet 2: " + val);\
            break;
        default:
            post("left inlet: " + val);
    }
}
```

inlets number

Number of inlets of the current object instance

The inlets property must be set in the global code to have any effect. If it isn't set, an object with one inlet will be created.

jsarguments string[] read-only

Access arguments typed into the js object when instantiated

The filename is `jsarguments[0]`, and the first typed-in argument is `jsarguments[1]`, and so on. The `jsarguments[]` array is available in global code and any function and it doesn't change unless the `js` object receives the `jsargs` message with new typed-in arguments.

Example

Creating an object with a variable number of outlets based on an argument typed into the `js` object

```
oulets = 0
if (jsarguments.length >= 2) {
    outlets = jsarguments[1];
}
if (!outlets) {
    oulets = 1;
}
```

max Max

read-only

Global Max instance

Gets a JavaScript representation of the "max" object (e.g. the recipient of `; max preempt 1` in a message box). This lets you send any message to the object that controls the Max application.

Example

```
max.preempt(1);
```

messagename string

read-only

Name of the message to the `js` object that invoked the method currently running

In global code, this is a `nil` value. This is generally useful only from within an `anything` function that will be called when no specific function name matches the message sent to the `js` object.

Example

Note the use of JavaScript bracket notation to specify a variable property

```
var stuff;
function anything(val) {
    if (arguments.length) {
        stuff[messagename] = val;
    }
}
```

outlets number

Number of outlets the `js` object should have

The outlets property must be set in the global code to have any effect. If it isn't set, and object with one outlet will be created.

Example

```
outlets = 2;
function anything() {
    outlet(0, "this is outlet 0");
    outlet(1, "this", "is", "outlet", "1");
}
```

Methods

anything

The `anything` function is called if no specific function is found to match the message symbol received by the `js` object

If you want to know the name of the message that invoked the function, use the `message` property. If you want to know what inlet received the message, use the `inlet` property.

```
anything(): void;
```

Example

```
function anything() {
    var val = arrayfromargs(messagename, arguments);
    post("anything() was called: " + val.toSource());
}
```

arrayfromargs

A utility for writing functions that take a variable number of arguments and/or those that can be called using various messages (like `anything`)

The function object has an `arguments` property that can be numerically indexed like an `Array` but is not an instance of `Array`. This means that you cannot call `Array` functions such as `sort()` on the `arguments` property or send the `arguments` property out of an outlet as a list of values. The `arrayfromargs` method will convert the arguments property to an `Array`, optionally with `message` as the zeroth element of the array. This message usage is useful for processing messages as though they are lists beginning with a symbol, as would be typical in your `anything` function.

```
arrayfromargs(arguments: object): string[];
```

Name	Type	Description
arguments	object	a property of the `arrayfromargs` function which holds all arguments passed to the function in an array-like container
Return Value	string[]	

Example 1

In this example, the arguments passed to the `js` object are sorted and sent out the first outlet.

```
function anything() {
    // messagename is a property of the `jsthis` object and
    // returns the name of the message which invoked the function
    var a = arrayfromargs(messagename, arguments);
    a.sort();
    outlet(0, a);
}
```

Example 2

In this example, any list sent to the `js` object is printed as an array.

```
function list() {
   var a = arrayfromargs(arguments);
   post("received list " + a + "\n");
}
```

arrayfromargs

A utility for writing functions that take a variable number of arguments and/or those that can be called using various messages (like `anything`)

The function object has an `arguments` property that can be numerically indexed like an `Array` but is not an instance of `Array`. This means that you cannot call `Array` functions such as `sort()` on the `arguments` property or send the `arguments` property out of an outlet as a list of values. The `arrayfromargs` method will convert the arguments property to an `Array`, optionally with `message` as the zeroth element of the array. This message usage is useful for processing messages as though they are lists beginning with a symbol, as would be typical in your `anything` function.

```
arrayfromargs(message: string, arguments: object): string[];
```

Name	Type	Description
message	string	the name of the message which triggered the function call; typically `messagename` is used here in the context of an `anything` method
arguments	object	a property of the `arrayfromargs` function which holds all arguments passed to the function in an array-like container
Return Value	string[]	

Example

In this example, the arguments passed to the `js` object are sorted and sent out the first outlet.

```
function anything() {
    // messagename is a property of the `jsthis` object and
    // returns the name of the message which invoked the function
    var a = arrayfromargs(messagename, arguments);
    a.sort();
    outlet(0, a);
}
```

assist

Set the patcher assist string for a designated inlet or outlet of a `js` object

This method is designed to be called from the assistance function, specified as an argument to jsthis.setinletassist() or jsthis.setoutletassist().

```
assist(...args: any[]): void;
```

Name	Type	Description
args	any[]	the assist string; if an array is supplied, the elements are concatenated

Example

```
outlets = 2;

function describe(n) {
    assist("This is outlet number: ", n);
}

setoutletassist(-1, describe);
```

bang

A function called when the `js` object receives a bang

```
bang(): void;
```

Example

```
function bang() {
    post("a bang was received\n");
}
```

declareattribute

Declare an attribute which can be set, queried, and optionally stored in the patcher file

If no getter or setter methods are specified, default ones will be used. These attributes can also be referenced by pattr.

```
declareattribute(attributeName: string, getterName?: string, setterName?:
string, embed?: boolean, options?: object): void;
```

Name	Type	Description
attributeName	string	the name of the attribute
optional getterName	string	the getter function name
optional setterName	string	the setter function name
optional embed	boolean	whether to embed on patcher save
optional options	object	a key/value dictinoary for additional options (only available in v8 engine)

Example 1

A simple attribute declaration where default getters and setters are used

```
var foo = 2;
declareattribute("foo");

function bang() {
    outlet(0, foo);
}
```

Example 2

A default getter and setter are used and the attribute is stored in the patcher file on save

```
var foo = 2;
declareattribute("foo", null, null, 1);

function bang() {
    outlet(0, foo);
}
```

Example 3

An explicit getter and setter are declared

```
var foo = 2;
declareattribute("foo", "getfoo", "setfoo");

function getfoo() {
    return foo;
}

function setfoo(v) {
    foo = v;
}

function bang() {
    outlet(0, foo);
}
```

Example 4

More examples using options object in v8 engine

```
// for any attribute that uses default getters and/or setters,
// it is important to declare the attribute using 'var' insetad of 'let'

var fluffy = 0;
declareattribute("fluffy", { style: "onoff", embed: 1, label: "Enable
Fluffiness" });

var bunny = 74;
declareattribute("bunny", { setter : "setbunny", min: -3, max: 101,
default: 74, type: "long" });

var bingo = "One";
declareattribute({ name: "bingo", style: "enum", enumvals: ["One", "Two",
"Three"] });

var bongo = [ 1, 0, 0, 1];
declareattribute({ name: "bongo", style: "rgba" });

var boppy = [ 74 , 75 ];
declareattribute("boppy", { min: -3, max: 101, default: [ 74, 75 ], type:
"long" });

// custom setter
function setbunny(v)
{
    bunny = v;
    post("I am setting bunny to: " + v + "\n");
}

// possible properties to pass in are:
// name : string
// getter : string
// setter : string
// embed : number/bool
// type: string ("long", "float", "symbol", "atom")
// size : number (for arrays...also implicit if default is an array)
// style : string ("onoff", "enum", "rgba")
// label : string
// category : string
// min : number
// max : number
// default : number, string, array
// invisible : number/bool
```

```
// steps : number// enumindex : array of strings// enumvals : array of
strings// paint : number/bool
```

embedmessage

The jsthis.embedmessage() method works only inside of the jsthis.save() function. It is used to specify the name of a function you want to be called when the `js` object containing the script is recreated.

```
embedmessage(functionName: string, args: number | number[] | string |
string[]): void;
```

Name	Type	Description
functionName	string	the name of a function to be called when jsthis.save() is recreated
args	number | number[] | string | string[]	arguments to pass to the named function

Example

In this example, when the `js` object containing this script is instantiated, the function `numchairs` will be called with an argument of 20 followed by the `numtables` function with an argument of `2`. Finally, the `roomname` function will be called with an argument of `"diningroom"`.

```
function save() {
    embedmessage("numchairs", 20);
    embedmessage("numtables", 2);
    embedmessage("roomname", "diningroom");
}

function numchairs(v) {
    // ...
}

function numtables(v) {
    // ...
}

function roomname(x) {
    // ...
}
```

getvalueof

Permit pattr and related objects to attach to and query an object's current value.

The value of an object returned can be a number, string, or an array of numbers and/or strings.

```
getvalueof(): any | any[];
```

Name	Type	Description
Return Value	any | any[]	

Example 1

```
myValue = 100;

function getvalueof() {
    return myValue;
}
```

Example 2

`getvalueof` works for other objects, not just `js` and `jsui`

```
var flonum = this.patcher.newobject("flonum");
flonum.setvalueof();
post("value: " + flonum.getvalueof() + "\n");
```

list

A function called when the `js` object receives a Max list (i.e. messages that begin with a number)

In implementing a list function, you'll probably want to use the `arguments` property so you can handle input of varying length.

```
list(): void;
```

Example

```
function list() {
    var val = arrayfromargs(messagename, arguments);
    post("list was called: " + val.toSource() + "\n");
}
```

loadbang

A function called when a patcher containing the `js` or `jsui` object is loaded

This function will not be called when you instantiate a new `js` or `jsui` object and add it to a patcher; it will only be called when a pre-existing patcher file containing a `js` object is loaded - in other words, at the same time that `loadbang` objects in a patcher are sending out bangs. You may wish to test the `loadbangdisabled` property of the Max object and do nothing in your loadbang function if it is true.

```
loadbang(): void;
```

Example

```
function loadbang() {
    post("loadbang was called\n");
}
```

msg_array

A function called when the `js` object receives a Max array, automatically converting the incoming Max array object to a native JS array object copy

This behavior is only available in the new v8 javascript engine objects. Alternatively, a function named "array" may defined to receive the name of the Max array object. This alternate strategy is also only available in the new v8 javascript engine.

```
msg_array(value: Array): void;
```

Name	Type	Description
value	Array	

Example

```
function msg_array(a) {
    if (a) {
        for (var i = 0; i < a.length; i++) {
            a[i] = a[i] + 1; // increment each array element by 1
        }
        outlet_array(0, a); // outlet and convert to Max array object
    }
}
```

msg_dictionary

A function called when the `js` object receives a Max dictionary, automatically converting the incoming Max dictionary object to a native JS object copy

The JS object will match the hierarchy and contents of the dictionary similar to if you serialized the dictionary to JSON and then parsed the JSON into a JS object. This behavior is only available in the new v8 javascript engine objects. Alternatively, a function named "dictionary" may defined to receive the name of the Max dictionary object. This alternate strategy may be used in the legacy js engine.

```
msg_dictionary(value: Object): void;
```

Name	Type	Description
value	Object	

Example

```
function msg_dictionary(d) {
    if (d) {
        d.x = "marks the spot"; // add a new key
        outlet_dictionary(0, d); // outlet and convert to a Max dictionary
object
    }
}
```

msg_float

A function called when the `js` object receives a float

If no jsthis.msg_int() method is defined, all numbers will be passed to the jsthis.msg_float() method and ints will be converted to floats before being used.

```
msg_float(value: number): void;
```

Name	Type	Description
value	number	

Example

```
function msg_float(v) {
    post(v);
}
```

msg_int

A function called when the `js` object receives an integer

If no jsthis.msg_float() method is defined, all numbers will be passed to the jsthis.msg_int() method and ints will be truncated to ints before being used.

```
msg_int(): void;
```

Example

```
function msg_int(v) {
    post(v);
}
```

msg_string

A function called when the `js` object receives a Max string, automatically converting the incoming Max dictionary object to a native JS string copy

This behavior is only available in the new v8 javascript engine objects. Alternatively, a function named "string" may defined to receive the name of the Max string object. This alternate strategy is

also only available in the new v8 javascript engine.

```
msg_string(value: string): void;
```

Name	Type	Description
value	string	

Example

```
function msg_string(s) {
    if (s) {
        s = s + " more text"; // append "more text"
        outlet_string(0, d); // outlet and convert to a Max string object
    }
}
```

notifyclients

Notify any clients (such as the pattr family of objects) that the object's current value has changed

Clients can then take appropriate action (such as sending a `js` instance the message `getvalueof` to invoke the jsthis.getvalueof() method if defined). The jsthis.notifyclients() method is useful for othat define jsthis.setvalueof() and jsthis.getvalueof() for pattr compatibility.

```
notifyclients(): void;
```

notifydeleted

A function called when the `js` / `jsui` object is freed

```
notifydeleted(): void;
```

outlet_array

Convert JS array to Max array and send through an outlet

This behavior is only available in the new v8 javascript engine objects. Alternatively, a JS array must be converted to a MaxArray instance and the message "array" followed by the MaxArray instance name may be sent out using the standard jsthis.outlet() method -- e.g. outlet(0, "array", myarray.name);

```
outlet_array(n: number, array: Array): void;
```

Name	Type	Description
n	number	the outlet number, 0-indexed
array	Array	JS array to send out of the outlet

outlet_dictionary

Convert JS object to Max dictionary and send through an outlet

The Max dictionary will match the hierarchy and contents of the JS object similar to if you serialized the JS object to JSON and then parsed the JSON into a Max dictionary. This behavior is only available in the new v8 javascript engine objects. Alternatively, a JS object must be converted to a Dict instance and the message "dictionary" followed by the Dict instance name may be sent out using the standard jsthis.outlet() method -- e.g. outlet(0, "dictionary", mydict.name);

```
outlet_dictionary(n: number, dictionary: Object): void;
```

Name	Type	Description
n	number	the outlet number, 0-indexed
dictionary	Object	JS Object to send out of the outlet

outlet_string

Convert JS string to Max string and send through an outlet

This behavior is only available in the new v8 javascript engine objects. Alternatively, a JS array must be converted to a MaxString instance and the message "string" followed by the MaxString instance name may be sent out using the standard jsthis.outlet() method -- e.g. outlet(0, "string", mystring.name);

```
outlet_string(n: number, string: String): void;
```

Name	Type	Description
n	number	the outlet number, 0-indexed
string	String	JS string to send out of the outlet

outlet

Send data through an outlet

If the outlet number if greater than the number of outlets, no output occurs.

If the argument to jsthis.outlet() is a JavaScript object, it is passed as the Max message `jsobject` which is the address of the object. When `jsobject` followed by a number is sent to a `js` object, it is parsed and checked to see if the number specifies the address of a valid JavaScript object. If so, the word `jsobject` disappears and the function sees only the JavaScript object reference.

If the argument to jsthis.outlet() is an array, it is unrolled one level and passed as a Max message or list (depending on whether the first element of the array is a number or string).

```
outlet(n: number, ...args: any[]): void;
```

Name	Type	Description
n	number	the outlet number, 0-indexed
args	any[]	anything to send out of the outlet

refresh

Copy the contents of this.sketch to the screen

This function is only available when using the jsui object. Call this function after drawing using the Sketch object to move the contents of sketch to the screen. Must be called after making any changes to the drawing in order to see those changes. If you're using MGraphics to draw instead, use MGraphics.redraw()

```
refresh(): void;
```

Example

Draw a ball at a randmo position when jsui gets a bang

```
function bang()
{
        with (Math) {
                with (sketch) {
                        shapeslice(vslices);
                        moveto((random()-0.5)*2,(random()-0.5)*2,
(random()-0.5)*2);
                        glcolor(random(),random(),random(),1);
                        sphere(random()*0.4);
                }
        }
        refresh();
}
```

save

Allow your script to embed state in a patcher file containing your `js` object and restore the state when the patcher is reloaded

Saving your state consists of storing a set of messages that your script will receive shortly after the `js` object containing it is recreated. These messages are stored using the jsthis.embedmessage() which only words inside your save function.

```
save(): void;
```

Example

When the patch containing the `js` object is saved, preserve the current value of a variable

```
var numcowbells = 1;
function cowbells(a) {
    numcowbells = 1;
}

function save() {
    embedmessage("cowbells", numcowbells);
}
```

setinletassist

Associates either a number, string, or function with a numbered inlet

If `-1` is passed as the inlet number, the object argument is used for all inlets. In order to produce any assistance text in the patcher window the assistance function needs to call the jsthis.assist() method. For an example use, see jsthis.setoutletassist(). The jsthis.setinletassist() and jsthis.setoutletassist() functions are best called in global code but can be called at any time. You can even replace the assistance function or string dynamically.

```
setinletassist(n: number, callback: Function): void;
```

Name	Type	Description
n	number	inlet number (0-indexed)
callback	Function	function which calls the jsthis.assist() method

setoutletassist

Associates either a number, string, or function with a numbered outlet

If `-1` is passed as the outlet number, the object argument is used for all inlets. In order to produce any assistance text in the patcher window the assistance function needs to call the jsthis.assist() method. The jsthis.setinletassist() and jsthis.setoutletassist() functions are best called in global code but can be called at any time. You can even replace the assistance function or string dynamically.

```
setoutletassist(n: number, callback: Function): void;
```

Name	Type	Description
n	number	outlet number (0-indexed)
callback	Function	function which calls the jsthis.assist() method

Example

```
outlets = 2;

function describe(n) {
    assist("This is outlet number: ", n);
}

setoutletassist(-1, describe);
```

setvalueof

Permit pattr and related objects to attach to and set an object's current value

Values passed will be of type `number` or `string`. For a value that consists of more than one `number` or `string`, the jsthis.setvalueof() method will receive multiple arguments. The `jsthis arrayfromargs` method is useful to handle values that can contain a variable number of elements.

```
setvalueof(...args: number[] | string[]): void;
```

Name	Type	Description
args	number[] | string[]	values to set

Example

`setvalueof` works for other objects, not just `js` and `jsui`

```
var flonum = this.patcher.newobject("flonum");
flonum.setvalueof();
post("value: " + flonum.getvalueof() + "\n");
```

enum LineStrokeStyleParameterNames

Stroke parameters for use with Sketch.beginstroke() in the "line" drawing style

Members

Member	Value	Description
alpha	`"alpha"`	May vary point to point. Value is specified as an alpha value. Useful if alpha is the only color channel which will vary throughout the path.
color	`"color"`	May vary point to point. Values are specified as red, green, blue, alpha.
order	`"order"`	Global for a given path. Value must be interpolation order. Default is 3, or bi-cubic interpolation.
slices	`"slices"`	Global for a given path. Number of slices for a curved section. Default is 20.

class LiveAPI

A means of communicating with the Live API from JavaScript.

For background information on this functionality, please see the Live API Overview and Live Object Model documents, as well as the Reference pages for live.path, live.object and live.observer objects, which provide the same basic functionality as the LiveAPI object, but from the Max patcher.

Technical note: you cannot use the LiveAPI object in JavaScript global code. Use the live.thisdevice object to determine when your Max Device has completely loaded (the object sends a bang from its left outlet when the Device is fully initialized, including the Live API).

Legacy note: previous versions of the LiveAPI object required the `jsthis` object's `this.patcher` property as the first argument. For backward-compatibility, this first argument is still supported, but is no longer necessary.

Beginning with release 6.0 of Max, it is no longer possible to configure JavaScript functions to run in the high-priority thread of Max's scheduler. The LiveAPI object cannot be created or used in the high-priority thread, so users should be sure to use the defer or deferlow objects to re-queue messages to the js object.

Example

```
var api = new LiveAPI(sample_callback, "live_set tracks 0");
if (!api) {
    post("no api object\n");
    return;
}
post("api.mode", api.mode ? "follows path" : "follows object", "\n");
post("api.id is", api.id, "\n");
post("api.path is", api.path, "\n");
post("api.children are", api.children, "\n");
post('api.getcount("devices")', api.getcount("devices"), "\n");

api.property = "mute";
post("api.property is", api.property, "\n");
post("type of", api.property, "is", api.proptype, "\n");

function sample_callback(args) {
    post("callback called with arguments:", args, "\n");
}
```

Constructors

```
new LiveAPI(callback?: Function, path?: string);
```

Constructs a new instance of the `LiveAPI` class

Parameter	Type	Description
optional callback	Function	a function to be called when the LiveAPI object refers to a new object in Live (if the LiveAPI object's path changes, for instance) or when an observed property changes
optional path	string	the object in Live pointed to by the LiveAPI object (e.g. `"live_set tracks 0 devices 0"`) or a valid LiveAPI object id

Properties

children string[]

An array of children of the object at the current path

id string

The id of the Live object referred to by the LiveAPI object. These ids are dynamic and awarded in realtime from the Live application, so should not be stored and used over multiple runs of Max for Live.

info string

read-only

A description of the object at the current path, including id, type, children, properties and functions

mode number

The follow mode of the LiveAPI object. 0 (default) means that LiveAPI follows the object referred to by the path, even if it is moved in the Live user interface.

For instance, consider a Live Set with two tracks, "Track 1" and "Track 2", left and right respectively. If your LiveAPI object's path is live_set tracks 0, the left-most track, it will refer to "Track 1". Should the position of "Track 1" change, such that it is now to the right of "Track 2", the LiveAPI object continues to refer to "Track 1". A mode of 1 means that LiveAPI updates the followed object based on its location in the Live user interface. In the above example, the LiveAPI object would always refer to the left-most track, updating its id when the object at that position in the user interface changes.

patcher Patcher

read-only

The patcher of the LiveAPI object, as passed into the constructor

path string

The path to the Live object referred to by the LiveAPI object.

These paths are dependent on the currently open Set in Live, but are otherwise stable: `live_set tracks 0 devices 0` will always refer to the first device of the first track of the open Live Set.

property string

The observed property, child or child-list of the object at the current path, if desired

For instance, if the LiveAPI object refers to "live_set tracks 1", setting the property to "mute" would cause changes to the "mute" property of the 2nd track to be reported to the callback function defined in the LiveAPI Constructor.

proptype string

read-only

The type of the currently observed property or child

The types of the properties and children are given in the Live Object Model.

type string

read-only

The type of the object at the current path

Please see the Live API Overview and Live Object Model documents for more information.

unquotedpath string

The path to the Live object referred to by the LiveAPI object, without any quoting (the path property contains a quoted path)

These paths are dependent on the currently open Set in Live, but are otherwise stable: live_set tracks 0 devices 0 will always refer to the first device of the first track of the open Live Set.

Methods

call

Calls the given function of the current object, optionally with a list of arguments.

```
call(fn: Function, ...arguments: any[]): void;
```

Name	Type	Description
fn	Function	a callback function
arguments	any[]	any arguments to the callback

get

Returns the value or list of values of the specified property of the current object.

```
get(property: string): number | number[];
```

Name	Type	Description
property	string	the object's property
Return Value	number | number[]	

getcount

The count of children of the object at the current path

```
getcount(child: string): number;
```

Name	Type	Description
child	string	the child to count children of
Return Value	number	

getstring

Returns the value or list of values of the specified property of the current object as a String object.

```
getstring(property: string): string | string[];
```

Name	Type	Description
property	string	the object's property
Return Value	string | string[]	

goto

Navigates to the path and causes the id of the object at that path out be sent to the callback function defined in the Constructor. If there is no object at the path, id 0 is sent.

```
goto(path: string): void;
```

Name	Type	Description
path	string	

set

Sets the value or list of values of the specified property of the current object.

```
set(property: string, value: any): void;
```

Name	Type	Description
property	string	the object's property to set
value	any	the new value or values of the property

class Max

Singleton Max object, controlling the Max environment.

Use the singleton instance of this object to control the Max environment. In the context of JavaScript executed in Max, the global `Max` singleton is always bound to an object named `max`.

Example 1

```
var arch = max.arch;
// The max object is already bound to the singleton Max object

post(arch);
// Prints the architecture of the current Max application.
```

In addition to the properties and methods described here, you can also use the global Max object to call methods as defined in Controlling Max with Messages.

Example 2

```
// Message box contents:
; max preempt 1

// JavaScript equivalent:
max.preempt(1);
```

Properties

apppath string

read-only

The pathname of the Max application

arch "arm64" | "x86" | "x64"

read-only

The architecture of the Max application

cmdkeydown number

read-only

Command key state

1 if the command (Macintosh) or control (Windows) key is currently held down.

ctrlkeydown number

read-only

Control key state

1 if the control key is currently held down.

frontpatcher Patcher

read-only

The Patcher object of the frontmost patcher window

The Patcher object of the frontmost patcher window, or a nil value if no patcher window is visible. You can traverse the list of open patcher windows with the next property of a Wind object.

isplugin number

read-only

Whether the js object is in a plugin

Will be 1 if the `js` object is in a plugin, and 0 otherwise. This will generally only be 1 if the `js` object is loaded as a Max device in a Max `vst~` object.

isruntime number

read-only

True if the current Max environment does not allow editing.

Returns 1 if the currently executing Max application environment does not allow editing, 0 if it does.

loadbangdisabled number

read-only

True if the user disabled loadbang.

1 if the user has disabled loadbang for the currently loading patch. If your object implements a loadbang method, it can test this property and choose to do nothing if it is true.

optionkeydown number

read-only

Option/alt key state

1 if the option (Macintosh) or alt (Windows) key is currently held down

os string

read-only

The name of the operating system

The name of the platform (e.g., “windows” or “macintosh”)

osversion string

read-only

The current OS version number

shiftkeydown string

read-only

Shift key state

1 if the shift key is currently held down

time number

read-only

Current scheduler time in milliseconds.

Will be a floating point value.

version string

read-only

Max application version number

Will be in string form, like "835"

Methods

getattr

Get the value of the named attribute

```
getattr(name: string): number | string | any[];
```

Name	Type	Description
name	string	the name of the attribute to retrieve
Return Value	number | string | any[]	the value of the attribute, as an array if the attribute value is a list

getattrnames

Available attributes of the Max global object

```
getattrnames(): string[];
```

Name	Type	Description
Return Value	string[]	the available attributes

getcolor

Get the value of the named color in the current theme

```
getcolor(name: string): number[4] | Dict;
```

Name	Type	Description
name	string	the name of the color to retrieve
Return Value	number[4] | Dict	the value of the color, possibly as a gradient dictionary

setattr

Set the value of the named attribute

```
setattr(name: string, value: any): void;
```

Name	Type	Description
name	string	the name of the attribute to set
value	any	the value of the attribute to set

class Maxobj

A JavaScript representation of a Max object in a patcher.

Perhaps the most powerful thing about a Maxobj is that you can send any message to a Maxobj that you can send to a Max object. For example, if you had a number box Maxobj and you wanted to set its value to 23 without outputting the value, you could do this:

```
n.set(23);
```

Note that certain words such as `int`, `float`, and `delete` are keywords in JavaScript, and you will need to use either array notation or the message method for such reserved words. For example:

```
n.message("int", 23);
n["int"](23);
```

Properties

background boolean

Whether the object is in the patchers background layer

boxtext string

read-only

The text contained in the object box (if present)

This property is only available in the new v8 javascript engine objects.

canhilite boolean

read-only

Whether the object can be selected for text entry

A number box would be one example of an object which would return `true`

colorindex number

> **This API is deprecated**
>
> If the object is set to use one of the standard 16 colors, the index of the color

hidden boolean

Whether the object is hidden in a locked patcher

ignoreclick boolean

Whether the object ignores clicks

js object

read-only

If the Maxobj refers to an object that is a `js` Max class, this returns the associated `jsthis` object

maxclass string

read-only

The Max class

This is different from the JavaScript class ("Maxobj") which is accessed via the standard class property.

nextobject Maxobj | undefined

read-only

The next object in the patcher's list of objects, otherwise nil

patchcords object

read-only

An object containing two arrays, `inputs`, and `outputs`, each of which may contain MaxobjConnection objects

patcher Patcher

read-only

The Patcher object that contains the Maxobj

rect [number, number, number, number]

The location of an object in a patcher (left, top, right, bottom)

When the object's rectangle is changed, it will move on screen if it's visible.

selected boolean

read-only

Whether the object is selected in an unlocked patcher window

valid boolean

read-only

Whether the Maxobj refers to a valid Max object

A Maxobj could eventually refer to an object that no longer exists if the underlying Max object is freed. The valid property can be used to test for this condition.

varname string

The patcher-specific name of the object as set in the inspector or via the Object Name... menu option

Methods

getattr

Get the value of an attribute

```
getattr(attrName: string): number | number[] | string;
```

Name	Type	Description
attrName	string	the attribute name
Return Value	number | number[] | string	

getattrattr

Get the value of an attribute's attribute

This method is only available in the new v8 javascript engine objects.

```
getattrattr(attrName: string, attrAttrName: string): number | number[] |
string;
```

Name	Type	Description
attrName	string	the attribute name
attrAttrName	string	the attribute name
Return Value	number | number[] | string	

getattrnames

Get all available attributes for the object

```
getattrnames(): string[];
```

Name	Type	Description
Return Value	string[]	

getboxattr

Get the value of the object's box attribute

```
getboxattr(attrName: string): number | number[] | string;
```

Name	Type	Description
attrName	string	the attribute name
Return Value	number | number[] | string	

getboxattrattr

Get the value of the object's box attribute's attribute

This method is only available in the new v8 javascript engine objects.

```
getboxattrattr(attrName: string, attrAttrName: string): number | number[]
| string;
```

Name	Type	Description
attrName	string	the attribute name
attrAttrName	string	the attribute name
Return Value	number | number[] | string	

getboxattrnames

Get the names of all available attributes for the object's box

```
getboxattrnames(): string[];
```

Name	Type	Description
Return Value	string[]	

help

Open a help file describing an object, if it exists

```
help(): void;
```

message

Send the object a message with any additional arguments

This is useful for sending messages to objects which dynamically dispatch messages with the `anything` message like `js`, `jsui`, `lcd`, and others.

```
message(message: string, ...args: any[]): void;
```

Name	Type	Description
message	string	the message to send
args	any[]	any arguments for the message

setattr

Set the value of an attribute

```
setattr(attrName: string, value: number | number[] | string): void;
```

Name	Type	Description
attrName	string	the attribute name
value	number | number[] | string	the new attribute value

setattrdefault

Set the value of an attribute to its default value (if a default value is defined)

This method is only available in the new v8 javascript engine objects.

```
setattrdefault(attrName: string): void;
```

Name	Type	Description
attrName	string	the attribute name

setboxattr

Set the value of a box attribute

```
setboxattr(attrName: string, value: number | number[] | string): void;
```

Name	Type	Description
attrName	string	the attribute name
value	number | number[] | string	the new attribute value

setboxattrdefault

Set the value of the object's box attribute to its default value (if a default value is defined)

This method is only available in the new v8 javascript engine objects.

```
setboxattrdefault(attrName: string): void;
```

Name	Type	Description
attrName	string	the attribute name

subpatcher

If the object contains a patcher, returns a Patcher, otherwise nil

```
subpatcher(index?: number): Patcher | undefined;
```

Name	Type	Description
optional index	number	an instance number (only used with `poly~`)
Return Value	Patcher | undefined	

understands

Get whether the object has an entry in its message list for a given string

If the entry is not a message that can be sent by a user within Max (i.e. it's a C-level "untyped" message), `false` is returned. This doesn't work for messages which are dynamically dispatched with the `anything` message, as is the case for instances of `js`, `jsui`, `lcd`, and others.

```
understands(message: string): boolean;
```

Name	Type	Description
message	string	an object message
Return Value	boolean	

class MaxobjConnection

A JavaScript representation of a patchcord connection.

Properties

dstobject Maxobj

read-only

The destination Maxobj

srcobject Maxobj

read-only

The source Maxobj

class MaxobjListener

A listener for changes in a Maxobj object.

The MaxobjListener object listens for changes to a Maxobj object's value, or changes to a specified attribute of a Maxobj object. When a change occurs, a user-specified function will be called. The object also provides methods for getting and setting the value of the observed value or attribute.

For convenience, the `MaxobjListener` object is a property of the MaxobjListenerData argument passed to the callback function. To access the `ParameterListener` from within its function, use MaxobjListenerData.listener.

Example

```
function valuechanged(data) {
    post("value changed!\n");
    if (data.attrname) {
        post("attrname: " + data.attrname + "\n");
    }
    post("new value: " + data.value + "\n");
}

var ob = this.patcher.getnamed("someobject");
var l = new MaxobjListener(ob, "patching_rect", valuechanged);
```

Constructors

```
new MaxobjListener(object: Maxobj, fn: Function);
```

Constructs a new instance of the `MaxobjListener` class

Without an attribute name provided, the listener will observe the value of the object itself. Not every Max object has an observable value -- objects compatible with the pattr family of Max objects can be observed in this fashion. Practically, that means nearly every UI object as well as a handful of normal Max box objects (including js, pattr and dict). Attributes can be observed for any Maxobj which has attributes.

Parameter	Type	Description
object	Maxobj	the object to attach a listener to
fn	Function	the callback function which takes a MaxobjListenerData as an argument

```
new MaxobjListener(object: Maxobj, attrName: string, fn: Function);
```

Create a MaxobjListener that observes a specific attribute

Parameter	Type	Description
object	Maxobj	the object to attach a listener to
attrName	string	the attribute to listen to
fn	Function	the callback function which takes a MaxobjListenerData as an argument

Properties

attrname string

read-only

An attribute to observe for changes, if desired

maxobj Maxobj

read-only

The Maxobj to observe

silent number

Whether to execute the callback function in response to calling MaxobjListener.setvalue() from this `MaxobjListener`

Methods

getvalue

Get the value of the Maxobj or its specified attribute

```
getvalue(): number | number[] | string;
```

Name	Type	Description
Return Value	number | number[] | string	

setvalue_silent

Set the value of a the Maxobj or its specified attribute, but don't execute the callback function

```
setvalue_silent(value: number): void;
```

Name	Type	Description
value	number	the new value

setvalue

Set the value of the Maxobj or its specified attribute

```
setvalue(value: any): void;
```

Name	Type	Description
value	any	the new value

class MaxobjListenerData

The argument provided to a MaxobjListener callback function.

Properties

attrname string | undefined

read-only

If the MaxobjListener is observing an attribute, the attributes name, otherwise undefined

listener MaxobjListener

read-only

The MaxobjListener which called the function

maxobject Maxobj

read-only

The Maxobj being observed

value number | number[] | string

read-only

The current value of the observed object or attribute

class MaxString

Bind a Max `string` object

Create a MaxString object when you want to bing to a Max `string` object, either because you want to fetch its value or when you want to modify its contents. To manipulate the contents of the string, get the value of the string using .stringify and then use regular JavaScript string functions.

Example 1

```
function string(str_name) {
        var max_str = new MaxString();
        max_str.name = str_name; // binds to the Max `string` by name
        var contents = max_str.stringify(); // read the value of the
string
}
```

Example 2

```
function lower(str_name) {
        var max_str = new MaxString();
        max_str.name = str_name; // binds to the Max `string` by name
        var js_str = max_str.stringify(); // get the value of the string
        js_str = js_str.toLowerCase(); // regular JavaScript string
functions
        max_str.parse(js_str); // set the string's new value
}
```

Constructors

```
new MaxString(initial_value: string?, ...attr_pairs: string?);
```

Create a new MaxString

You can set the name of the string either by passing the name after the argument "name", or you can set the .name property after creating the MaxString objects.

Parameter	Type	Description
initial_value	string?	initial value
attr_pairs	string?	usually the string "name" followed by the name of the string

Example

```
// These both create a string that binds to the same named string.
var my_str = new MaxString("initial_value", "@name", "fred");

var my_str2 = new MaxString("initial_value");
my_str2.name = "fred";
```

Properties

name string

Get and set the name of the MaxString

Will bind to an existing Max string with the same name

Methods

parse

Update the value of the MaxString

```
parse(value: any);
```

Name	Type	Description
value	any	value to parse into a string

stringify

Get the current value of the MaxString as a string

```
stringify(): string;
```

Name	Type	Description
Return Value	string	The current value of the MaxString

function messnamed

Sends a message to the named Max object.

A named Max object is an object associated with a global symbol (not an object with a patcher-specific name). For example, Max receive objects are bound to global symbols. The code in the first example sends the message bang to the named object flower.

The arguments passed to the object can be individual arguments or an array.

```
export declare function messnamed(objectName: string, selector: string,
args: any): void;
```

Name	Type	Description
objectName	string	the name of the object to send the message to
selector	string	the name of the object method to call
args	any	the arguments to pass to the object method

Example 1

```
messnamed("flower", "bang");
```

Example 2

```
messnamed("flower", "count", 1, 2, 3);
messnamed("flower", "count", [1, 2, 3]);
```

class MGraphics

Drawing context for rendering shapes and images.

The MGraphics object combines a virtual drawing canvas with functions to manage drawing to that canvas. In enables you to draw simple shapes, complex paths, text, and images.

Most of the time, you'll work with the global MGraphics object, always stored in a global variable called mgraphics, in order to perform custom drawing of a jsui object, or in a jspainter file. In that case, make sure to call the MGraphics.init() method somewhere in global scope:

```
// Initialize mgraphics
mgraphics.init();

// Now you can use mgraphics in your custom drawing code
function paint() {
    mgraphics.rectangle(50, 0, 200, 100);
    mgraphics.fill();
}
```

It's also possible to create your own MGraphics object in order to draw to an offsceen . In this case, there is no need to call MGraphics.init().

Constructors

```
new MGraphics(width: number, height: number);
```

Creates a new MGraphics instance

Create a MGraphics context, including a buffer of offscreen memory to to render to, which can be used as a drawing context for creating saved or persistent images. An MGraphics instance can also be copied into an object by passing the MGraphics instance to the Image constructor.

Parameter	Type	Description
width	number	Width of the offscreen drawing area
height	number	Height of the offscreen drawing area

Properties

autofill 1 | 0

Enable or disable automatic fill after stroke.

With autofill enabled, any call to MGraphics.stroke() will also fill the current path automatically.

autopaint 1 | 0

Turns on/off painting of the global MGraphics object.

See MGraphics.autosketch for technical details about managing autosketch and autopaint. In most cases, use the MGraphics.init() method to enable custom drawing with mgraphics.

autosketch 1 | 0

Turns on/off painting with the global sketch object.

In the context of a jsui or jspainter object, the object draws itself using two layers: sketch and mgraphics. By default, only the sketch layer renders, and the mgraphics layer is unused. If the mgraphics layer is enabled, then the object will call into the user-defined paint() function to draw

the mgraphics layer on top of the sketch layer. Calling MGraphics.init() will set autosketch to 0 and autopaint to 1, disabling the sketch layer and displaying only custom drawing in the mgraphics layer. In most cases, this is the easiest way to implement custom drawing.

relative_coords 1 | 0

Enable or disable the use of relative coordinates.

When disabled, the origin [0, 0] will be the top-left, and [width, height] will be the bottom-right, where width and height are the dimensions of MGraphics.size. When relative_coords is enabled, the origin [0, 0] will be in the center of the drawing area. The y-coordinate -1 will always be the top, and 1 will always be the bottom. The x-coordinate of the left edge will depend on the aspect ratio of the drawing area. If the width of the drawing area is x times the height, then the left edge will be -x and the right edge will be x.

size [number, number]

read-only

Get the current size of the MGraphics canvas

If you're using the global mgraphics instance as part of custom drawing code for jsui or with jspainter, then this will be the size of the object being redrawn.

Methods

append_path

Appends a stored path to the current path at the current end point.

```
append_path(path: MGraphicsPathHandle);
```

Name	Type	Description
path	MGraphicsPathHandle	A path handle as returned from MGraphics.copy_path().

arc_negative

Add a circular, counter-clockwise arc to the current path.

```
arc_negative(xc: number, yc: number, radius: number, angle1: number,
angle2: number);
```

Name	Type	Description
xc	number	x coordinate of the center of the arc
yc	number	y coordinate of the center of the arc
radius	number	radius of the arc
angle1	number	start angle for the arc segment in radians
angle2	number	end angle for the arc segment in radians

arc

Add a circular, clockwise arc to the current path.

```
arc(xc: number, yc: number, radius: number, angle1: number, angle2:
number);
```

Name	Type	Description
xc	number	x coordinate of the center of the arc
yc	number	y coordinate of the center of the arc
radius	number	radius of the arc
angle1	number	start angle for the arc segment in radians
angle2	number	end angle for the arc segment in radians

close_path

Close the current path.

Adds a line from the current point to the starting point of the path, which closes it.

```
close_path();
```

copy_path

Returns a copy of the current path.

Returns a copy of the current path to be stored and reused later.

```
copy_path(): MGraphicsPathHandle;
```

Name	Type	Description
Return Value	MGraphicsPathHandle	

curve_to

Add a cubic Bezier spline to the current path.

```
curve_to(x1: number, y1: number, x2: number, y2: number, x3: number, y3:
number);
```

Name	Type	Description
x1	number	x coordinate of the first control point
y1	number	y coordinate of the first control point
x2	number	x coordinate of the second control point
y2	number	y coordinate of the second control point
x3	number	x coordinate of the destination point
y3	number	y coordinate of the destination point

device_to_user

Convert a point in device space to user space.

The inverse of MGraphics.user_to_device().

```
device_to_user(position: [number, number]): [number, number];
```

Name	Type	Description
position	[number, number]	A two-element array containing the x, y coordinate of a point in device space.
Return Value	[number, number]	The equivalent point in user space.

ellipse

Add a closed ellipse to the current path.

```
ellipse(x: number, y: number, width: number, height: number);
```

Name	Type	Description
x	number	x coordinate of the top-left of the rectangle containing the ellipse
y	number	y coordinate of the top-left of the rectangle containing the ellipse
width	number	width of the ellipse
height	number	height of the ellipse

fill_extent

Get the enclosing rectangle of the current path.

```
fill_extent(): [number, number, number, number];
```

Name	Type	Description
Return Value	[number, number, number, number]	An array containing the top, left, bottom, and right extremes of the rectangle.

fill_preserve_with_alpha

Fill and preserve the current path with alpha override.

Fill the current path, but override the alpha value of the current color with a new alpha channel value. This lets you change transparency without resetting the color values. Do not discard the path afterwards.

```
fill_preserve_with_alpha(alpha: number);
```

Name	Type	Description
alpha	number	the opacity between 0 and 1

fill_preserve

Draw and preserve the current path.

Fill the current path, using the current MGraphics context settings including color. Do not discard the path afterwards, which can be useful, for example, when you want to draw the stroke and fill with different colors.

```
fill_preserve();
```

fill_with_alpha

Fill the current path with alpha override.

Fill the current path, but override the alpha value of the current color with a new alpha channel value. This lets you change transparency without resetting the color values. The path is discarded afterwards.

```
fill_with_alpha(alpha: number);
```

Name	Type	Description
alpha	number	the opacity between 0 and 1

fill

Fill the current path.

Fill the current path, using the current MGraphics context settings including color. The path is discarded afterwards.

```
fill();
```

font_extents

Get ascent, descent, and height for font.

Returns the ascent, descent, and height for the current font. Ascent measures the distance from the text baseline to the tallest glyph. Descent measures the distance from the text baseline to the lowest point on any glyph under the baseline. Height is simply ascent plut descent.

```
font_extents(): [number, number, number];
```

Name	Type	Description
Return Value	[number, number, number]	An array containing font ascent, descent, and height.

get_current_point

The current drawing position.

This is the point at which any new additions to the current path would begin. It is also the end point of the current path, and if you close the current path, this point will be connected back to the start of the the path. Most functions that add to the path will move this point.

```
get_current_point(): [number, number];
```

Name	Type	Description
Return Value	[number, number]	

get_line_cap

Get the current line cap.

```
get_line_cap(): "butt" | "round" | "square";
```

Name	Type	Description
Return Value	"butt" | "round" | "square"	

get_line_join

Get the current line join.

```
get_line_join(): "miter" | "round" | "bevel";
```

Name	Type	Description
Return Value	"miter" | "round" | "bevel"	

get_line_width

Get the current line width.

```
get_line_width(): number;
```

Name	Type	Description
Return Value	number	

get_matrix

Retrieve the current transformation matrix.

```
get_matrix(): MGraphicsMatrixHandle;
```

Name	Type	Description
Return Value	MGraphicsMatrixHandle	

getfontlist

Get a list of installed fonts.

Get a list of all fonts installed on your system.

```
getfontlist(): string[];
```

Name	Type	Description
Return Value	string[]	

identity_matrix

Reset the transform matrix to identity (no transformation).

```
identity_matrix();
```

image_surface_draw

Draw an image into the current surface.

Place an image (typically stored as an Image object) into the current surface. The drawing is placed at the top-left of the drawing context, changeable using a transform matrix or translate function. You can also choose the section of the image to draw using four optional arguments that describe a rectangle taken from the image.

```
image_surface_draw(image: Image, source_rect?: [number, number, number,
number]);
```

Name	Type	Description
image	Image	to draw
optional source_rect	[number, number, number, number]	(optional) section of the image to draw

in_fill

Determine if a point is within the current path.

Assuming the current path is fillable, tests if the given position is contained in the current path.

```
in_fill(position: [number, number]): 1 | 0;
```

Name	Type	Description
position	[number, number]	The position to test
Return Value	1 | 0	1 if the point is in the path, and 0 otherwise.

init

Initialize the mgraphics system.

Initialize the mgraphics system. Call this somewhere in global scope (usually near the top of your JavaScript file) in order to use mgraphics in your custom drawing code.

```
init(): void;
```

line_to

Add a line segment to the current path.

```
line_to(x: number, y: number);
```

Name	Type	Description
x	number	x coordinate of the destination point
y	number	y coordinate of the destination point

move_to

Move the current point to a new location.

Move the current point to a new location, and begin a new subpath.

```
move_to(x: number, y: number);
```

Name	Type	Description
x	number	x coordinate of the new location
y	number	y coordinate of the new location

ovalarc

Add an ellipical arc to the current path.

```
ovalarc(xc: number, yc: number, radiusx: number, radiusy: number, angle1:
number, angle2: number);
```

Name	Type	Description
xc	number	x coordinate of the center of the arc
yc	number	y coordinate of the center of the arc
radiusx	number	horizontal radius of the arc.
radiusy	number	vertical radius of the arc.
angle1	number	start angle for the arc segment in radians
angle2	number	end angle for the arc segment in radians

path_roundcorners

Round the corners of the current path.

Modify the current path by rounding the corners to the radius provided, or as close as possible depending on the path's angle.

```
path_roundcorners(radius: number);
```

Name	Type	Description
radius	number	the radius of the rounded corners

pattern_create_for_surface

Create a pattern using an image.

Create a pattern using an image for the background. Repeating patterns depends on the extend value set using the Pattern.set_extend() function

```
pattern_create_for_surface(image: Image): Pattern;
```

Name	Type	Description
image	Image	
Return Value	Pattern	

pattern_create_linear

Create a linear gradient.

```
pattern_create_linear(x1: number, y1: number, x2: number, y2: number):
Pattern;
```

Name	Type	Description
x1	number	x coordinate of first color stop
y1	number	y coordinate of first color stop
x2	number	x coordinate of second color stop
y2	number	y coordinate of second color stop
Return Value	Pattern	

pattern_create_radial

Create a radial gradient.

```
pattern_create_radial(x1: number, y1: number, rad1: number, x2: number,
y2: number, rad2: number): Pattern;
```

Name	Type	Description
x1	number	x coordinate of first color stop
y1	number	y coordinate of first color stop
rad1	number	radius of first color stop
x2	number	x coordinate of second color stop
y2	number	y coordinate of second color stop
rad2	number	radius of second color stop

pattern_create_rgba

Create a solid color pattern.

```
pattern_create_rgba(red: number, green: number, blue: number, alpha:
number): Pattern;
```

Name	Type	Description
red	number	
green	number	
blue	number	
alpha	number	
Return Value	Pattern	

pop_group

Completes a path execution group and returns the result.

Complete a path execution group, returning the results as a Pattern.

```
pop_group(): Pattern;
```

Name	Type	Description
Return Value	Pattern	

push_group

Define a starting point for a path execution group.

Define a starting point for a path execution group. This group can be used for creating an image from a set of path functions without actually drawing the results to the screen.

```
push_group();
```

rectangle_rounded

Add a closed rounded-rectangle to the current path.

```
rectangle_rounded(x: number, y: number, width: number, height: number,
ovalwidth: number, ovalheight: number);
```

Name	Type	Description
x	number	x coordinate of the top-left of the rectangle
y	number	y coordinate of the top-left of the rectangle
width	number	width of the rectangle
height	number	height of the rectangle
ovalwidth	number	width of the oval used for the rounded corners
ovalheight	number	height of the oval used for the rounded corners

rectangle

Add a closed rectangle to the current path

```
rectangle(x: number, y: number, width: number, height: number);
```

Name	Type	Description
x	number	x coordinate of the top-left of the rectangle
y	number	y coordinate of the top-left of the rectangle
width	number	width of the rectangle
height	number	height of the rectangle

redraw

Request that the current display area be redrawn.

Request that the current display area be redrawn. Max will then call your custom paint() function as part of the next available drawing loop. You should never call your custom paint() function directly.

```
redraw(): void;
```

rel_curve_to

Add a cubic Bezier spline relative to the current point.

Add a cubic Bezier spline to the current path. The arguments to this function are coordinates relative to the current point.

```
rel_curve_to(x1: number, y1: number, x2: number, y2: number, x3: number,
y3: number);
```

Name	Type	Description
x1	number	x coordinate of the first control point
y1	number	y coordinate of the first control point
x2	number	x coordinate of the second control point
y2	number	y coordinate of the second control point
x3	number	x coordinate of the destination point
y3	number	y coordinate of the destination point

rel_line_to

Add a line segment relative to the current point.

Add a line segment to the current path. The arguments to this function are coordinates relative to the current point.

```
rel_line_to(x: number, y: number);
```

Name	Type	Description
x	number	x coordinate of the destination point
y	number	y coordinate of the destination point

rel_move_to

Move the current point relative to the current point.

Move the current point to a new location, and begin a new subpath. The arguments to this function are relative to the current point.

```
rel_move_to(x: number, y: number);
```

Name	Type	Description
x	number	x coordinate of the new location
y	number	y coordinate of the new location

restore

Pop the current MGraphics state.

Pop the most recent saved MGraphics state off the stack and apply it. See MGraphics.save().

```
restore(): void;
```

rotate

Adds a rotation to the current transform matrix.

```
rotate(angle: number);
```

Name	Type	Description
angle	number	the rotation angle in radians (counter-clockwise)

save

Push the current MGraphics state.

Push the current MGraphics state, including stroke and fill color, the current matrix transform, line style, and font style, onto the state stack. After making changes, call MGraphics.restore() to return to the saved state.

```
save(): void;
```

scale_source_rgba

Modifies the color transform by adding a scale.

Modifies the color transform by adding a scale. When drawing, MGraphics will multiple the current source color by this scale before drawing. This can be useful if you want to modify the source color in some way without losing the source color. Calling this function again will scale the color further. There is no way to reset the color transform without using MGraphics.save() and MGraphics.restore().

```
scale_source_rgba(sc_red: number, sc_green: number, sc_blue: number,
sc_alpha: number);
```

Name	Type	Description
sc_red	number	red channel color scaling

sc_green	number	green channel color scaling
sc_blue	number	blue channel color scaling
sc_alpha	number	alpha channel color scaling

scale

Add a scale to the current transform matrix.

Add a scale to the current transform matrix. This will stretch or squish the drawing horizontally or vertially. Affects line widths.

```
scale(scale_x: number, scale_y: number);
```

Name	Type	Description
scale_x	number	
scale_y	number	

select_font_face

Sets the current font face by name

```
select_font_face(fontname: string);
```

Name	Type	Description
fontname	string	

set_font_size

Set the current font size

Set the current font size. Floating point and integers are both accepted.

```
set_font_size(fontsize: number);
```

Name	Type	Description
fontsize	number	

set_line_cap

Set the drawing style for the end point of a line.

```
set_line_cap(cap: "butt" | "round" | "square");
```

Name	Type	Description
cap	"butt" | "round" | "square"	The desired drawing style

set_line_join

Set the appearance of the connection point between lines.

```
set_line_join(join: "miter" | "round" | "bevel");
```

Name	Type	Description
join	"miter" | "round" | "bevel"	The desired connection style.

set_line_width

Set the current line width.

Set the width of path lines drawn using the MGraphics.stroke() function. The interpretation of this value is dependent on the coordinate system in use, set with MGraphics.relative_coords

```
set_line_width(width: number);
```

Name	Type	Description
width	number	Line width

set_matrix

Set the current transform matrix directly.

```
set_matrix(xx: number, xy: number, yx: number, yy: number, x0: number, y0:
number);
```

Name	Type	Description
xx	number	x scaling support
xy	number	rotational warping
yx	number	rotational warping
yy	number	y scaling support
x0	number	x translation
y0	number	y translation

set_source_rgb

Set the color used for drawing.

Set the color used for drawing, and set the opacity to 1 (fully opaque).

```
set_source_rgb(red: number, green: number, blue: number);
```

Name	Type	Description
red	number	
green	number	
blue	number	

set_source_rgba

Set the color and alpha used for drawing.

Set the color and alpha used for drawing. Values are between 0 and 1.

```
set_source_rgba(red: number, green: number, blue: number, alpha: number);
```

Name	Type	Description
red	number	
green	number	
blue	number	
alpha	number	

set_source_surface

Sets an image to use as a drawing source

```
set_source_surface(image: Image, dx?: number, dy?: number);
```

Name	Type	Description
image	Image	to draw
optional dx	number	translation in x
optional dy	number	translation in y

set_source

Set the source pattern.

Sets the pattern to be used for the next MGraphics.fill() call.

```
set_source(pattern: Pattern);
```

Name	Type	Description
pattern	Pattern	

show_text

Draw text without a path.

Draws the text at the current location, using the current font face and font size. This does not create a path or modify the current path. Matrix transforms will not affect this drawing.

```
show_text(text_to_display: string);
```

Name	Type	Description
text_to_display	string	text to draw

stroke_preserve_with_alpha

Draw and preserve the outline of the current path with alpha override.

Draw the outline of the current path, but override the alpha value of the current color with a new alpha channel value. This lets you change transparency without resetting the color values. Do not discard the path afterwards.

```
stroke_preserve_with_alpha(alpha: number);
```

Name	Type	Description
alpha	number	the opacity between 0 and 1

stroke_preserve

Draw and preserve the outline of the current path.

Draw the outline of the current path, using the current settings for color, line width, etc. Do not discard the path afterwards, which can be useful, for example, when you want to draw the stroke and fill with different colors.

```
stroke_preserve();
```

stroke_with_alpha

Draw the outline of the current path with alpha override.

Draw the outline of the current path, but override the alpha value of the current color with a new alpha channel value. This lets you change transparency without resetting the color values. The path is discarded afterwards.

```
stroke_with_alpha(alpha: number);
```

Name	Type	Description
alpha	number	the opacity between 0 and 1

stroke

Draw the outline of the current path.

Draw the outline of the current path, using the current settings for color, line width, etc. The path is discarded afterwards.

```
stroke();
```

svg_render

Render an SVG image.

Draw an SVG image in the current graphics context. The SVG source can be either a filename, or a raw SVG string, or an MGraphicsSVG object. Optional x, y, width, and height arguments determine the destination rect into which the SVG will be rendered.

```
svg_render(source: MGraphicsSVG | string, x?: number, y?: number, width?:
number, height?: number, opacity?: number);
```

Name	Type	Description
source	MGraphicsSVG | string	SVG object, filename, or raw SVG string
optional x	number	destination origin x coordinate

optional width	number	destination width
optional height	number	destination height
optional opacity	number	opacity (0-1)

text_measure

Measure the bounds of text

Returns an array contaning the width and height of the given text, as it would appear if rendered in the current font face and font size.

```
text_measure(text: string): [number, number];
```

Name	Type	Description
text	string	Text to measure
Return Value	[number, number]	Text width and height

text_path

Create a path from text

Creates a path from the given text, using the current font face and font size. Once the text is in the path, it can be transformed just like an ordinary path.

```
text_path(text_to_display: string);
```

Name	Type	Description
text_to_display	string	build a path for this text

transform

Modify the current transform matrix by multiplying.

Modify the current transform matrix by multiplying. Useful if you already have a transform matrix that you want to include in the current matrix transform.

```
transform(xx: number, xy: number, yx: number, yy: number, x0: number, y0:
number);
```

Name	Type	Description
xx	number	x scaling support
xy	number	rotational warping
yx	number	rotational warping
yy	number	y scaling support
x0	number	x translation
y0	number	y translation

translate_source_rgba

Modifies the color transform by adding an offset.

Modifies the color transform by adding an offset. When drawing, MGraphics will add this offset to the current source color before drawing. This can be useful if you want to modify the source color

in some way without losing the source color. Calling this function again will increase or decrease the color offset. There is no way to reset the color transform without using MGraphics.save() and MGraphics.restore().

```
translate_source_rgba(t_red: number, t_green: number, t_blue: number,
t_alpha: number);
```

Name	Type	Description
t_red	number	red channel color offset
t_green	number	green channel color offset
t_blue	number	blue channel color offset
t_alpha	number	alpha channel color offset

translate

Adds a translation to the current transform matrix.

```
translate(t_x: number, t_y: number);
```

Name	Type	Description
t_x	number	horizontal translation
t_y	number	vertical translation

user_to_device

Convert a user space point to device space.

Convert a point in user space, in which drawing occurs, to device space, the coordinate system after matrix transforms have been applied. If MGraphics were an oil painting, user space would be the position of a point on the canvas, and device space would be where that same point appears in a photograph taken of the painting.

```
user_to_device(position: [number, number]): [number, number];
```

Name	Type	Description
position	[number, number]	A two-element array containing the x, y coordinate of a point in user space. *
Return Value	[number, number]	The equivalent point in device space.

type MGraphicsMatrixHandle

Opaque handle for an MGraphics transformation matrix.

```
export declare type MGraphicsMatrixHandle = any;
```

type MGraphicsPathHandle

Opaque handle for an MGraphics path.

```
export declare type MGraphicsPathHandle = any;
```

class MGraphicsSVG

SVG object handle

The function MGraphics.svg_render() can draw an SVG file directly from a filename. However, drawing from an instance of MGraphicsSVG can be more efficient, since the SVG file does not need to be reloaded. You can also use the functions MGraphicsSVG.mapcolor() and MGraphicsSVG.mapreset() to map the colors in the SVG source to a new color, which can be useful to render the same image multiple times with different states.

Example

```
// Using an SVG file in Max's search path
var my_svg = new MGraphicsSVG("icon.svg");
var disabled = false;

function paint() {

                my_svg.mapreset();
                if (disabled) {
                        my_svg.mapcolor(
                                [0, 0, 0, 1.0],
                                [0.5, 0.5, 0.5, 1.0]
                        )
                }

                mgraphics.svg_render(my_svg);
}
```

Constructors

```
new MGraphicsSVG(filename: string);
```

Create an SVG object whose colors can be remapped

Parameter	Type	Description
filename	string	Name of the SVG file, in Max's search path

Methods

mapcolor

Remap source SVG color

```
mapcolor(source_color: [number, number, number, number], map_color:
[number, number, number, number]): void;
```

Name	Type	Description
source_color	[number, number, number, number]	Origin color in RGBA format
map_color	[number, number, number, number]	Destination color in RGBA format

mapreset

Clear all color remappings

```
mapreset(): void;
```

class ParameterInfoProvider

Provides a list of named parameter objects within a patcher hierarchy as well as information about specific parameter objects. It can also notify when parameter objects are added or removed from a patcher hierarchy.

ParameterInfoProvider is not yet supported in the v8 engine.

Example

```
function paramschanged(data) {
    post("something was added or removed - getting a new list\n");
    if (data.added.length) {
        post(data.added.join(", ") + "added\n");
    }
    if (data.removed.length) {
        post(data.removed.join(", "), + "removed\n");
    }
}
pip = new ParameterInfoProvider(paramschanged);
```

Constructors

```
new ParameterInfoProvider(fn?: Function);
```

Constructs a new instance of the `ParameterInfoProvider` class

Parameter	Type	Description
optional fn	Function	a callback function to execute when receiving parameter change notifications which takes a ParameterInfoProviderData as an argument

Methods

getinfo

Get parameter info like its type and range

The exact contents of the object will vary depending on the type of the parameter and will need to be enumerated. The object always contains a property `maxobj` which is the Maxobj for the Max object hosting the parameter.

```
getinfo(paramName: string): any[];
```

Name	Type	Description
paramName	string	the parameter to find info for
Return Value	any[]	

Example

```
var pip = new ParameterInfoProvider();
var info = pip.getinfo("live.dial");
for (var i in info) {
    post(i + ": " + info[i] + "\n");
}
```

for a default `live.dial` object, this will print

```
js: longname: live.dial
js: shortname: live.dial
js: scriptname: live.dial
js: type: float
js: visibility: automated
js: min: 0
js: max: 127
js: exponent: 1
js: maxobject: [object Maxobj]
```

getnames

Get a list of parameter objects' names from the patcher hierarchy

```
getnames(): string[];
```

Name	Type	Description
Return Value	string[]	

class ParameterInfoProviderData

The argument to the ParameterInfoProvider's callback function

Properties

added string[]

read-only

The names of any added parameters

provider ParameterInfoProvider

read-only

The ParameterInfoProvider which called the function

removed string[]

read-only

The names of any removed parameters

class ParameterListener

A listener for changes in named parameters.

The `ParameterListener` listens for changes to the value of a named Max parameter. When a change occurs, a user-specified callback function will be called. The object also provides methods for getting and setting the value of the observed parameter.

For convenience, the `ParameterListener` object is a property of the ParameterListenerData argument passed to the callback function. To access the `ParameterListener` from within its function, use ParameterListenerData.listener.

ParameterListener is not yet supported in the v8 engine.

Example

```
function valuechanged(data) {
    post("parameter value changed: " + data.name + "\n");
    post("new value: " + data.value + "\n");
}

var l = new ParameterListener("myParameter", valuechanged);
```

Constructors

```
new ParameterListener(paramName: string, fn: Function);
```

Constructs a new instance of the `ParameterListener` class

Parameter	Type	Description
paramName	string	the parameter name
fn	Function	the callback function which takes a ParameterListenerData as an argument

Properties

name string

read-only

The name of the parameter to observe

silent number

Whether to execute the callback function in response to calling ParameterListener.setvalue() from this `ParameterListener`

Methods

getvalue

Get the value of a parameter

```
getvalue(): number | number[] | string;
```

Name	Type	Description
Return Value	number | number[] | string	

setvalue_silent

Set the value of a parameter, but don't execute the callback function

```
setvalue_silent(value: number): void;
```

Name	Type	Description
value	number	the new parameter value

setvalue

Set the value of a parameter

```
setvalue(value: any): void;
```

Name	Type	Description
value	any	the new parameter value

class ParameterListenerData

The argument provided to a ParameterListener callback function.

Properties

listener ParameterListener

read-only

The ParameterListener which called the function

name string

read-only

The name of the changed parameter

value number | number[] | string

read-only

The current value of the parameter

class Patcher

A JavaScript representation of a Max patcher.

You can find, create, modify, and iterate through objects within a patcher, send messages to a patcher that you would use with the thispatcher object, and more.

There are currently three ways to get a Patcher:

1. Use the constructor method

2. Access the patcher property of `jsthis` (e.g. `this.patcher`)

3. Use the Maxobj.subpatcher() method

Any message to a patcher that you can send in Max (via the thispatcher object), you can send in JavaScript.

Example

```
p = this.patcher;
p.fullscreen(1) // make the patcher take up the whole screen
p.dirty() // make an editable patcher dirty
```

Constructors

```
new Patcher();
```

Constructs a new instance of the `Patcher` class with default window coordinates (100, 100, 400, 400)

```
new Patcher(left: number, top: number, bottom: number, right: number);
```

Constructs a new instance of the `Patcher` class

Parameter	Type	Description
left	number	x-coordinate of the new top-left corner
top	number	y-coordinate of the new top-left corner
bottom	number	y-coordinate of the new bottom-right corner
right	number	x-coordinate of the new bottom-right corner

Properties

accentcolor number[4]

read-only

The accent color in the current style

bgcolor number[4]

read-only

The background color in the current style

bgfillcolor Dict

read-only

The object background color in the current style

box Maxobj

read-only

If the patcher is a subpatcher, the `box` property returns the Maxobj that contains it.

Example

To traverse up to the top-level patcher:

```
var prev = 0;
var owner = this.patcher.box;
while (owner) {
    prev = owner;
    owner = owner.patcher.box;
}
if (prev) post("top patcher is", prev.patcher.name)
```

bubble_bgcolor number[4]

read-only

Bubble background color in the current style

bubble_outlinecolor number[4]

read-only

Bubble outline color in the current style

clearcolor number[4]

read-only

Clear color in the current style

color number[4]

read-only

The object color in the current style

count number

read-only

Number of objects in the patcher

darkcolor number[4]

read-only

Dark color in the current style

editing_bgcolor number[4]

read-only

Unlocked patcher background color in the current style

elementcolor number[4]

read-only

The element color in the current style

filepath string

read-only

The patcher's file path on disk

firstobject Maxobj

read-only

If the patcher contains objects, this is the first one in its list. You can iterate through all objects in a patcher using the Maxobj.nextobject property.

lightcolor number[4]

read-only

Light color in the current style

locked_bgcolor number[4]

read-only

Locked patcher background color in the current style

locked boolean

Whether the patcher is locked

This property is read-only in the runtime version of Max.

maxclass string

read-only

Returns "patcher"

name string

The patcher's name which is its window title (without any brackets that appear for subpatchers)

parentclass string

read-only

Get the Max class name of the parent object if `this.patcher` is a subpatcher, or a nil value if this is a top-level patcher

Example

```
function bang() {
    var pclass = this.patcher.parentclass;
    if (pclass) {
        post("The parent patcher class is " + pclass);
    } else {
        post("This is a top-level patcher");
    }
}
```

parentpatcher Patcher | undefined

read-only

Get the parent patcher if `this.patcher` is a subpatcher, otherwise a nil value

patchlinecolor number[4]

read-only

Patch cord color in the current style

scrolloffset [number, number]

The x-y coordinate array for the scroll offset of a patcher window

scrollorigin [number, number]

The x-y coordinate array of the patcher's fixed origin

selectioncolor number[4]

read-only

The selection color in the current style

stripecolor number[4]

read-only

Stripe color in the current style

syntax_attrargcolor number[4]

read-only

Syntax color for attribute arguments in the current style

syntax_attributecolor number[4]

read-only

Syntax color for object attributes in the current style

syntax_objargcolor number[4]

read-only

Syntax color for object arguments in the current style

syntax_objectcolor number[4]

read-only

Syntax color for object names in the current style

textcolor_inverse number[4]

read-only

The inverse text color in the current style

textcolor number[4]

read-only

The text color in the current style

wind Wind

read-only

Get the Wind associated with the patcher

Methods

apply

For all objects in a patcher, call a given function with each object's Maxobj as an argument

Does not recurse into subpatchers.

```
apply(fn: Function): void;
```

Name	Type	Description
fn	Function	A callback function

Example

The following prints the name of each object's class:

```
function printobj(obj) {
    post(obj.maxclass);
}
this.patcher.apply(printobj);
```

applydeep

For all objects in a patcher, recursing into subpatchers, call a given function with each object's Maxobj as an argument

```
applydeep(fn: Function): void;
```

Name	Type	Description
fn	Function	A callback function

applydeepif

For all objects in a patcher, recursing into subpatchers, call a given function with each object's Maxobj as an argument if a test function returns `true`

```
applydeepif(apply_fn: Function, test_fn: Function): void;
```

Name	Type	Description
apply_fn	Function	
test_fn	Function	

applyif

For all objects in a patcher, call a given function with each object's Maxobj as an argument if a test function returns `true`

Does not recurse into subpatchers.

```
applyif(applyFn: Function, testFn: Function): void;
```

Name	Type	Description
applyFn	Function	A callback function which takes a Maxobj and runs if `testFn` returns `true`
testFn	Function	A function which takes a Maxobj as an argument and returns a boolean

Example

The following prints the name of each object's class if the object is hidden

```
function printhidden(obj) {
    post(obj.maxclass);
}
function ishidden(obj) {
    return obj.hidden;
}
this.patcher.applyif(printhidden, ishidden);
```

bringtofront

Move an object to the front of the current layer (background or foreground)

You can change the layer by setting the Maxobj.background property.

```
bringtofront(object: Maxobj): void;
```

Name	Type	Description
object	Maxobj	the object to move

connect

Connect two `Maxobj` objects in a patcher

Indices for the outlet and inlet arguments start at 0 for the leftmost inlet or outlet.

```
connect(fromObj: Maxobj, outlet: number, toObj: Maxobj, inlet: number):
void;
```

Name	Type	Description
fromObj	Maxobj	Source object
outlet	number	Index of outlet from source object
toObj	Maxobj	Destination object
inlet	number	Index of inlet of destination object

Example

```
var p = this.patcher;
var a = p.newobject("toggle", 122, 90, 15, 0);
var b = p.newobject("toggle", 122, 140, 15, 0);
p.connect(a, 0, b, 0);
```

disconnect

Disconnect two connected `Maxobj` objects in a patcher

Indices for the outlet and inlet arguments start at 0 for the leftmost inlet or outlet.

```
disconnect(fromObj: Maxobj, outlet: number, toObj: Maxobj, inlet: number):
void;
```

Name	Type	Description
fromObj	Maxobj	Source object
outlet	number	Index of outlet from source object
toObj	Maxobj	Destination object
inlet	number	Index of inlet of destination object

Example

```
var p = this.patcher;
var a = p.newobject("toggle", 122, 90, 15, 0);
var b = p.newobject("toggle", 122, 140, 15, 0);
p.connect(a, 0, b, 0);
p.disconnect(a, 0, b, 0);
```

getattr

Get the value of a specified patcher attribute

```
getattr(attrname: string): string[];
```

Name	Type	Description
attrname	string	the attribute name
Return Value	string[]	

getattrattr

Get the value of a specified patcher attribute's attribute

This method is only available in the new v8 javascript engine objects.

```
getattrattr(attrName: string, attrAttrName: string): number | number[] |
string;
```

Name	Type	Description
attrName	string	the attribute name
attrAttrName	string	the attribute name
Return Value	number | number[] | string	

getattrnames

Get an array of all available attributes for the patcher

```
getattrnames(): string[];
```

Name	Type	Description
Return Value	string[]	

getlogical

Collect all objects in a patcher which, when passed to a test function, cause that function to return true

```
getlogical(testFn: Function): Maxobj[] | undefined;
```

Name	Type	Description
testFn	Function	A function which takes a Maxobj as an argument and returns a boolean
Return Value	Maxobj[] | undefined	

Example

```
function logical(a) {
    if (a) {
        return true;
    } else {
        return false;
    }
}

function loadbang() {
    // uses the return value as an array
    var found = patcher.getlogical(logical);
    if (found && found.length) {
        for (var i = 0; i < found.length; i++) {
            post(found[i].maxclass + ": " + found[i].rect + "\n");
        }
    }
}

function bang() {
    loadbang();
}
```

getnamed

Get the first object found in a patcher with a given name

The name is the local varname specified via the Object Name... menu or the `varname` property in the Inspector.

```
getnamed(name: string): Maxobj;
```

Name	Type	Description
name	string	The name of the object to retrieve

Return Value	Maxobj

hiddenconnect

Connect two `Maxobj` objects in a patcher with a hidden patch cord

Indices for the outlet and inlet arguments start at 0 for the leftmost inlet or outlet.

```
hiddenconnect(fromObj: Maxobj, outlet: number, toObj: Maxobj, inlet:
number): void;
```

Name	Type	Description
fromObj	Maxobj	Source object
outlet	number	Index of outlet from source object
toObj	Maxobj	Destination object
inlet	number	Index of inlet of destination object

Example

```
var p = this.patcher;
var a = p.newobject("toggle", 122, 90, 15, 0);
var b = p.newobject("toggle", 122, 140, 15, 0);
p.hiddenconnect(a, 0, b, 0);
```

newdefault

Create a new object at a specified location

```
newdefault(left: number, top: number, classname: string, ...args: any[]):
Maxobj;
```

Name	Type	Description
left	number	the x-coordinate of the new object's top-left corner
top	number	the y-coordinate of the new object's top-left corner
classname	string	the classname of the Max object to create
args	any[]	arguments to pass to the Max object
Return Value	Maxobj	

Example 1

```
var obj = patcher.newdefault(122, 90, "toggle");
```

Example 2

The `newdefault` method also accepts additional arguments for non UI objects that represent the created object's typed-in arguments.

```
var obj = patcher.newdefault(122, 90, "pack", "rgb", 255, 128, 64);
```

newobject

Create a new Max object

```
newobject(classname: string, ...args: any[]): Maxobj;
```

Name	Type	Description
classname	string	the classname of the Maxobj to create
args	any[]	any arguments to pass to the Maxobj
Return Value	Maxobj	

Example

```
a = patcher.newobject("toggle", 122, 90, 15, 0);
```

remove

Remove a Maxobj from the patcher

```
remove(object: Maxobj): void;
```

Name	Type	Description
object	Maxobj	The Maxobj to remove

sendtoback

Send an object to the back of the current layer (background or foreground)

You can change the layer by setting the Maxobj.background property.

```
sendtoback(object: Maxobj): void;
```

Name	Type	Description
object	Maxobj	the object to move

setattr

Set the value of a specified patcher attribute

```
setattr(attrname: string, value: any): void;
```

Name	Type	Description
attrname	string	the attribute name
value	any	the value of the attribute

setattrdefault

Set the value of a specified patcher attribute to its default value (if a default value is defined)

This method is only available in the new v8 javascript engine objects.

```
setattrdefault(attrName: string): void;
```

Name	Type	Description
attrName	string	the attribute name

class Pattern

Pattern object for drawing gradients and patterns.

The MGraphics object can create Pattern objects with functions like MGraphics.pattern_create_linear(), MGraphics.pattern_create_radial(), and MGraphics.pattern_create_rgba(). Generally, you use this Pattern object to add color stops and transformations to customize the pattern, then set that pattern as a source before filling a path.

Example

```
function paint() {
                // create a path
                mgraphics.rectangle(50, 0, 200, 100);

                // create a gradient pattern to fill the shape
                var tmp =
mgraphics.pattern_create_radial(20.,20.,20.,100.,100.,30.);
                tmp.add_color_stop_rgba(0.,1.,0.,0.,1.);
                tmp.add_color_stop_rgba(1.,0.,1.,0.,1.);

                // set the gradient as a source
                mgraphics.set_source(tmp);

                // fill the shape
                mgraphics.fill();
}
```

Methods

add_color_stop_rgba

Add a color stop to a pattern.

Sets a color stop for the pattern. Linear and radial gradients will fade continuously between these color values.

```
add_color_stop_rgba(index: number, red: number, green: number, blue:
number, alpha: number);
```

Name	Type	Description
index	number	The index of the color stop

red	number	The red component of the color stop
green	number	The green component of the color stop
blue	number	The blue component of the color stop
alpha	number	The alpha component of the color stop

get_extend

Get the extend value of the pattern

The extend value determines how the pattern will be drawn to fill extra space. "none" will draw nothing, "reflect" will invert the pattern and continue to draw it as if reflected, "repeat" will start the pattern over and continue to draw, and "pad" will take the last color of the pattern and extend it outward.

```
get_extend(): "none" | "reflect" | "repeat" | "pad";
```

Name	Type	Description
Return Value	"none" | "reflect" | "repeat" | "pad"	

get_type

Get the pattern type

Get the type of the pattern. Values are 0-solid, 1-surface, 2-linear, 3-gradient.

```
get_type(): 0 | 1 | 2 | 3;
```

Name	Type	Description
Return Value	0 | 1 | 2 | 3	

identity_matrix

Reset the transform matrix to identity (no transformation).

```
identity_matrix();
```

rotate

Adds a rotation to the pattern

```
rotate(angle: number);
```

Name	Type	Description
angle	number	the rotation angle in radians (counter-clockwise)

scale

Scale the pattern horizontally and vertically

```
scale(scale_x: number, scale_y: number);
```

Name	Type	Description
scale_x	number	horizontal scaling
scale_y	number	vertical scaling

set_extend

Set the extend value of the pattern

```
set_extend(extend_value: "none" | "reflect" | "repeat" | "pad");
```

Name	Type	Description
extend_value	"none" | "reflect" | "repeat" | "pad"	The extend value to set

set_matrix

Set the transform matrix for the pattern directly.

```
set_matrix(xx: number, xy: number, yx: number, yy: number, x0: number, y0:
number);
```

Name	Type	Description
xx	number	x scaling support
xy	number	rotational warping
yx	number	rotational warping

x0	number	x translation
y0	number	y translation

translate

Translate the current pattern

```
translate(t_x: number, t_y: number);
```

Name	Type	Description
t_x	number	horizontal translation
t_y	number	vertical translation

class PolyBuffer

Bind to a Max `polybuffer~` object.

The PolyBuffer object in JavaScript is a companion to the polybuffer~ object in Max. Through it, you can to access samples and metadata for the polybuffer~ object with the given name.

Constructors

```
new PolyBuffer(name: string);
```

Constructs a new instance of the `PolyBuffer` class

Parameter	Type	Description
name	string	name of the Max `polybuffer~` to bind to.

Properties

count string

read-only

Number of `buffer~` objects in the `polybuffer~`

name string

read-only

Name of the Max `polybuffer~`

size number

read-only

Memory size used by the `polybuffer~` in bytes

Methods

append

Add a sound file to the `polybuffer~`

```
append(soundfilePath?: string): void;
```

Name	Type	Description
optional soundfilePath	string	sound file path to load; if none provided, a dialog will appear

appendempty

Add an empty buffer~ with specified length and channel count

```
appendempty(duration: number, channels: number): void;
```

Name	Type	Description
duration	number	the duration in milliseconds
channels	number	the number of channels

clear

Delete every buffer~

```
clear(): void;
```

dump

Get info about a `polybuffer~`

```
dump(): [number, string, string, number, number, number];
```

Name	Type	Description
Return Value	[number, string, string, number, number, number]	- an array containing the index, name, path, duration, channel, and sample rate of `buffer~` s in the `polybuffer~`

getbufferlist

Get the names of `buffer~` s in the `polybuffer~`

```
getbufferlist(): string[];
```

Name	Type	Description
Return Value	string[]	

getshortname

Get every `buffer~` name followed by the sound file name (without extensions)

```
getshortname(): string[];
```

Name	Type	Description
Return Value	string[]	

open

Open the `polybuffer~` object's window to see information about the buffers

```
open(): void;
```

print

Post the `polybuffer~` s contents to the Max window

The content printed are the number of items in the `polybuffer~` and the shortname and filenames of each buffer in the `polybuffer~`.

```
print(): void;
```

readfolder

Load multiple sound files from the specified folder

```
readfolder(folderPath?: string): void;
```

Name	Type	Description
optional folderPath	string	folder to read; if none provided, a dialog will appear

send

Send messages to `buffer~` objects in the `polybuffer~`

```
send(index: number, message: any): void;
```

Name	Type	Description
index	number	the `buffer~` index (1-indexed); an index of 0 sends the message to every `buffer~`
message	any	the message to send

Example

The following example binds to a `polybuffer~` named `"mypolybuffer"` and clears the second `buffer~` if there is one.

```
var pb = new PolyBuffer("mypolybuffer");
pb.send(2, "clear");
```

wclose

Close the window editor

```
wclose(): void;
```

writefolder

Write every buffer~ to a fild in a folder

```
writefolder(folderPath?: string): void;
```

Name	Type	Description
optional folderPath	string	folder to read; if none provided, a dialog will appear

function post

Prints a representation of the arguments in the Max window.

If post() has no arguments, it prints starting on the next line. Otherwise it prints the input on the current line separated by spaces. Arrays are unrolled to one level as with jsthis.outlet().

```
export declare function post(...args: any): void;
```

Name	Type	Description
args	any	one or more arguments to print

Example

```
var a = new Array(900, 1000, 1100);
post(1, 2, 3, "violet", a);
post();
post(4, 5, 6);

// prints the following to the Max console:
// 1 2 3 violet 900 1000 1100
// 4 5 6
```

If you want a line break in the middle of a call to post() you can use "\n" within a string. Also, post() begins a new line if the last person to write to the Max window was not the js object.

If you try to print a JavaScript class instance, you will see jsobject printed to the console.

class Sketch

Interface to an OpenGL-backed drawing context

Every custom UI made with jsui or JSPainter has access to a default Sketch object bound to the global variable "sketch". Use this object to render to the OpenGL-backed drawing context available to all UI objects. Often this is the only instance of the Sketch object that you will use. If you want to render sprites, have multiple layers of images, or create alpha channels, you can construct new instances of the Sketch object.

Constructors

```
new Sketch(width?: number, height?: number);
```

Create a new Sketch instance

Parameter	Type	Description
optional width	number	width, leave undefined to use the default
optional height	number	height, leave undefined to use the default

Methods

beginstroke

Begin definition of a stroked path

```
beginstroke(stroke_style: "basic2d" | "line");
```

Name	Type	Description
stroke_style	"basic2d" | "line"	the stroke style to use

circle

Draw a circle or an arc

Draws a filled circle with radius specified by the radius argument at the current drawing position. If theta_start and theta_end are specified, then an arc will be drawn instead of a full circle. Affected by shapeorient, shapeslice, and shapeprim values.

```
circle(radius: number, theta_start?: number, theta_end?: number);
```

Name	Type	Description
radius	number	radius of the circle
optional theta_start	number	start angle in degrees
optional theta_end	number	end angle in degrees

copypixels

Copy pixels from one object to the current sketch

Copies pixels from the source object to the location specified by the destination_x and destination_y arguments. The initial x and y offset into the source and size of the rectangle copied can be speified by the source_x, source_y, width and height arguments. If these are not present, an x and y offset of zero and width and height equal to the source image is assumed. No scaling of pixels is supported. If blending is enabled in the destination sketch object, alpha blending will be performed and the current alpha color will also be applied globally. he copypixels method is much faster than obtaining the equivalent result using glbindtexture() to texture a plane, and is the recommended means of drawing images when scaling and rotation are not required.

```
copypixels(source_obj: Sketch | Image, destination_x: number,
destination_y: number, source_x?: number, source_y?: number, width?:
number, height?: number);
```

Name	Type	Description
source_obj	Sketch | Image	the source object to copy pixels from
destination_x	number	x coordinate of the destination
destination_y	number	y coordinate of the destination
optional source_x	number	x coordinate of the source
optional source_y	number	y coordinate of the source
optional width	number	width
optional height	number	height

cube

Draw a cube

The cube will have width = 2 * scale_x, height = 2 * scale_y, and depth = 2 * scale_z, and will be centered at the current drawing position. By default, scale_y and scale_z will be equal to scale_x. Affected by shapeorient, shapeslice, and shapeprim values.

```
cube(scale_x: number, scale_y?: number, scale_z?: number);
```

Name	Type	Description
scale_x	number	half width
optional scale_y	number	half height
optional scale_z	number	half depth

cylinder

Draw a cylinder or cylindrical arc

Draws a cylinder with top radius specified by the radius1 argument, bottom radius specified by the radius2 argument, length specified by the mag argument, and center point at the current drawing position. If the theta_start and theta_end arguments are specified, then a cylindrical wedge will be drawn instead of a full cylinder. Affected by shapeorient, shapeslice, and shapeprim values.

```
cylinder(radius1: number, radius2: number, mag: number, theta_start?:
number, theta_end?: number);
```

Name	Type	Description
radius1	number	radius of one end of the cylinder

radius2	number	radius of the other end of the cylinder
mag	number	height of the cylinder
optional theta_start	number	start angle in degrees
optional theta_end	number	end angle in degrees

default2d

Set the default graphics state for 2d rendering

The default2d method is a simple way to set the graphics state to default properties useful for 2D graphics. It is called everytime your object is resized if default2d() has been called more recently than default3d(). It is essentially equivalent to the following set of calls:

```
default2d();
```

Example

```
with (sketch) {
        glpolygonmode("front_and_back", "fill")
        glpointsize(1)
        gllinewidth(1)
        gldisable("depth_test")
        gldisable("fog")
        glcolor(0, 0, 0, 1)
        glshademodel("smooth")
        gldisable("lighting")
        gldisable("normalize")
        gldisable("texture")
        glmatrixmode("projection")
        glloadidentity()
        glortho(-aspect, aspect, -1, 1, -1, 100)
        glmatrixmode("modelview")
        glloadidentity()
        glulookat(0, 0, 2, 0, 0, 0, 0, 0, 1)
        glclearcolor(1, 1, 1, 1)
        glclear()
        glenable("blend")
        glblendfunc("src_alpha", "one_minus_src_alpha")
}
```

default3d

Set the default graphics state for 3d rendering

The default3d method is a simple way to set the graphics state to default properties useful for 3D graphics. It is called everytime the jsui object is resized if default3d() has been called more recently than default2d(). It is essentially equivalent to the following set of calls:

```
default3d();
```

Example

```
with (sketch) {
        glpolygonmode("front_and_back", "fill")
        glpointsize(1)
        gllinewidth(1)
        glenable("depth_test")
        glenable("fog")
        glcolor(0, 0, 0, 1)
        glshademodel("smooth")
        gllightmodel("two_side", "true")
        glenable("lighting")
        glenable("light0")
        glenable("normalize")
        gldisable("texture")
        glmatrixmode("projection")
        glloadidentity()
        gluperspective(default_lens_angle, aspect, 0.1, 100)
        glmatrixmode("modelview")
        glloadidentity()
        glulookat(0, 0, 2, 0, 0, 0, 0, 0, 1)
        glclearcolor(1, 1, 1, 1)
        glclear()
        glenable("blend")
        glblendfunc("src_alpha", "one_minus_src_alpha")
}
```

depthatpixel

Get the depth at a given pixel

Returns the depth value associated with the currently rendered pixel at a given absolute screen coordinate.

```
depthatpixel(x: number, y: number): number;
```

Name	Type	Description
x	number	screen x coordinate
y	number	screen y coordinate
Return Value	number	

ellipse

Draw an ellipse or elliptical arc

Draws a filled ellipse with radii specified by the radius1 and radius2 arguments. If theta_start and theta_end are specified, then an arc will be drawn instead of a full ellipse. Affected by shapeorient, shapeslice, and shapeprim values.

```
ellipse(radius1: number, radius2: number, theta_start?: number,
theta_end?: number);
```

Name	Type	Description
radius1	number	radius of the first axis
radius2	number	radius of the second axis
optional theta_start	number	start angle in degrees
optional theta_end	number	end angle in degrees

endstroke

End definition of a path and render it

```
endstroke();
```

font

Set the current font

```
font(font_name: string);
```

Name	Type	Description
font_name	string	name of the font

fontsize

Set the font size in points

```
fontsize(size: number);
```

Name	Type	Description
size	number	size of the font

framecircle

Draw a framed circle or arc

Draws a framed circle with radius specified by the radius argument at the current drawing position. If theta_start and theta_end are specified, then an arc will be drawn instead of a full circle. Affected by shapeorient, shapeslice, and shapeprim values.

```
framecircle(radius: number, theta_start?: number, theta_end?: number);
```

Name	Type	Description
radius	number	radius of the circle
optional theta_start	number	start angle in degrees
optional theta_end	number	end angle in degrees

frameellipse

Draw a framed ellipse or elliptical arc

Draws a framed ellipse with radii specified by the radius1 and radius2 arguments. If theta_start and theta_end are specified, then an arc will be drawn instead of a full ellipse. Affected by shapeorient, shapeslice, and shapeprim values.

```
frameellipse(radius1: number, radius2: number, theta_start?: number,
theta_end?: number);
```

Name	Type	Description
radius1	number	radius of the first axis
radius2	number	radius of the second axis
optional theta_start	number	start angle in degrees

framequad

Draw a framed quadrilateral

After this method has been called, the drawing position is updated to the location specified by the x4, y4, and z4 arguments.

```
framequad(x1: number, y1: number, z1: number, x2: number, y2: number, z2:
number, x3: number, y3: number, z3: number, x4: number, y4: number, z4:
number);
```

Name	Type	Description
x1	number	x coordinate of the first point
y1	number	y coordinate of the first point
z1	number	z coordinate of the first point
x2	number	x coordinate of the second point
y2	number	y coordinate of the second point
z2	number	z coordinate of the second point
x3	number	x coordinate of the third point
y3	number	y coordinate of the third point
z3	number	z coordinate of the third point
x4	number	x coordinate of the fourth point
y4	number	y coordinate of the fourth point
z4	number	z coordinate of the fourth point

frametri

Draw a framed triangle

After this method has been called, the drawing position is updated to the location specified by the x3, y3, and z3 arguments.

```
frametri(x1: number, y1: number, z1: number, x2: number, y2: number, z2:
number, x3: number, y3: number, z3: number);
```

Name	Type	Description
x1	number	x coordinate of the first point
y1	number	y coordinate of the first point
z1	number	z coordinate of the first point
x2	number	x coordinate of the second point
y2	number	y coordinate of the second point
z2	number	z coordinate of the second point
x3	number	x coordinate of the third point
y3	number	y coordinate of the third point
z3	number	z coordinate of the third point

freepeer

Free the native C peer

Frees data from the native C peer (created when making a Sketch object), which is not considered by the JavaScript garbage collector, and may consume lots of memory until the garbage collector decides to run based JS allocated memory. Once called, the Sketch object is not available for any other use. It's not necessary to call this function, as the memory will be freed eventually, but you can call it whenever you're done with your Sketch object.

```
freepeer();
```

getpixel

Get the pixel data at a given point

Returns an array containing the pixel value at the specified location. This array is ordered RGBA, i.e. array element 0 is red, 1, green, 2, blue, 3 alpha. Color values are floating point numbers in the range 0.-1.

```
getpixel(x: number, y: number): [number, number, number, number];
```

Name	Type	Description
x	number	x coordinate
y	number	y coordinate
Return Value	[number, number, number, number]	

gettextinfo

Get the rendered size of text

Returns an array containing the width and height of the given string in absolute screen coordinates, taking into account the current font and fontsize.

```
gettextinfo(text: string): [number, number];
```

Name	Type	Description
text	string	text to measure
Return Value	[number, number]	

glbegin

Begin drawing using low level OpenGL functions

The low level OpenGL function calls (all beginning with gl) are thin wrappers around direct calls to the graphics engine. Typically, use these function calls between calls to Sketch.glbegin() and Sketch.glend(). For many of these functions, look up the documentation for the OpenGL function with the same (or very similar) name.

```
glbegin(prim_type: DrawingPrimitiveType);
```

Name	Type	Description
prim_type	DrawingPrimitiveType	the drawing primitive to use

Example

```
var sx = 1.0;
var sy = 1.0;
var tx = 0.0;
var ty = 0.0;
function draw() {
        // refers to the global "sketch" object
 sketch.glclear();

        sketch.glcolor(1, 0, 0);
        sketch.glbegin("lines");

 // draw x axis
        glvertex(Math.min(tx + sx * -1, -1),  ty, 0);
        glvertex(Math.max(tx + sx * 1, 1),  ty, 0);

 // draw y axis
        glvertex(tx,  Math.min(ty + sy * -1, -1), 0);
        glvertex(tx,  Math.max(ty + sy * 1, 1), 0);
}
```

glbindtexture

Apply the given texture to subsequent drawing calls

Note: this method also calls glenable(texture)

```
glbindtexture(image: Image);
```

Name	Type	Description
image	Image	the image to use as a texture

glblendfunc

```
glblendfunc(src_func: string, dst_func: string);
```

Name	Type	Description
src_func	string	source function
dst_func	string	destination function

glclear

Clear the drawing context

```
glclear();
```

glclearcolor

Set the color to fill the context with using Sketch.glclear()

```
glclearcolor(red: number, green: number, blue: number, alpha: number = 1);
```

Name	Type	Description
red	number	red (0-1 range)
green	number	green (0-1 range)
blue	number	blue (0-1 range)
optional alpha	number	alpha (0-1 range)

glclearcolor

```
glclearcolor(colors: number[]);
```

Name	Type	Description
colors	number[]	

glcleardepth

Set the depth to fill the context with using Sketch.glclear()

```
glcleardepth(depth: number);
```

Name	Type	Description
depth	number	depth (0-1 range)

glclipplane

```
glclipplane(plane: number, coeff1: number, coeff2: number, coeff3: number,
coeff4: number);
```

Name	Type	Description
plane	number	

coeff1	number
coeff2	number
coeff3	number
coeff4	number

glclipplane

```
glclipplane(planeValues: number[]);
```

Name	Type	Description
planeValues	number[]	

glcolor

Set the color for subsequent drawing calls

```
glcolor(red: number, green: number, blue: number, alpha: number = 1);
```

Name	Type	Description
red	number	red (0-1 range)
green	number	green (0-1 range)
blue	number	blue (0-1 range)
optional alpha	number	alpha (0-1 range)

glcolor

```
glcolor(colors: number[]);
```

Name	Type	Description
colors	number[]	

glcolormask

```
glcolormask(red: number, green: number, blue: number, alpha: number = 1);
```

Name	Type	Description
red	number	
green	number	
blue	number	
optional alpha	number	

glcolormask

```
glcolormask(colors: number[]);
```

Name	Type	Description
colors	number[]	

glcolormaterial

```
glcolormaterial(face: number, mode: number);
```

Name	Type	Description
face	number	
mode	number	

glcullface

```
glcullface(face: number);
```

Name	Type	Description
face	number	

gldepthmask

```
gldepthmask(onoff: number);
```

Name	Type	Description
onoff	number	

gldepthrange

```
gldepthrange(near: number, far: number);
```

Name	Type	Description
near	number	
far	number	

gldisable

Disable a drawing capaility.

Usually "blend", "line_smooth", or "texture"

```
gldisable(capability: string);
```

Name	Type	Description
capability	string	the capability to disable

gldrawpixels

```
gldrawpixels(image: Image);
```

Name	Type	Description
image	Image	

gledgeflag

```
gledgeflag(onoff: number);
```

Name	Type	Description
onoff	number	

glenable

Enable a drawing capaility.

Usually "blend", "line_smooth", or "texture"

```
glenable(capability: string);
```

Name	Type	Description
capability	string	the capability to enable

glend

```
glend();
```

glfinish

```
glfinish();
```

glflush

```
glflush();
```

glfog

```
glfog(parameter_name: string, ...values: number);
```

Name	Type	Description
parameter_name	string	
values	number	

glfrustrum

```
glfrustrum(left: number, right: number, bottom: number, top: number, near:
number, far: number);
```

Name	Type	Description
left	number	
right	number	
bottom	number	
top	number	
near	number	
far	number	

glfrustrum

```
glfrustrum(frustrumValues: number[]);
```

Name	Type	Description
frustrumValues	number[]	

glhint

```
glhint(target: string, mode: number);
```

Name	Type	Description
target	string	
mode	number	

gllight

```
gllight(light: string, parameter_name: string, ...values: number);
```

Name	Type	Description
light	string	
parameter_name	string	
values	number	

gllightmodel

```
gllightmodel(light: string, model: number);
```

Name	Type	Description
light	string	
model	number	

gllinestipple

```
gllinestipple(factor: any, bit_pattern: any);
```

Name	Type	Description
factor	any	
bit_pattern	any	

gllinewidth

```
gllinewidth(width: number);
```

Name	Type	Description
width	number	

glloadidentity

Load the identity matrix

```
glloadidentity();
```

glloadmatrix

```
glloadmatrix(matrix_array: number[]);
```

Name	Type	Description
matrix_array	number[]	

gllogicop

```
gllogicop(op: number);
```

Name	Type	Description
op	number	

glmaterial

```
glmaterial();
```

glmatrixmode

```
glmatrixmode(mode: string);
```

Name	Type	Description
mode	string	

glmultmatrix

```
glmultmatrix(matrix_array: number[]);
```

Name	Type	Description
matrix_array	number[]	

glnormal

```
glnormal(x: number, y: number, z: number);
```

Name	Type	Description
x	number	
y	number	
z	number	

glortho

```
glortho(left: number, right: number, bottom: number, top: number, near:
number, far: number);
```

Name	Type	Description
left	number	
right	number	
bottom	number	

top	number
near	number
far	number

glortho

```
glortho(orthoValues: number[]);
```

Name	Type	Description
orthoValues	number[]	

glpointsize

```
glpointsize(size: number);
```

Name	Type	Description
size	number	

glpolygonmode

```
glpolygonmode(face: number, mode: number);
```

Name	Type	Description
face	number	
mode	number	

glpolygonoffset

```
glpolygonoffset(factor: number, units: number);
```

Name	Type	Description
factor	number	
units	number	

glpopattrib

```
glpopattrib();
```

glpopmatrix

```
glpopmatrix();
```

glpushattrib

```
glpushattrib();
```

glpushmatrix

```
glpushmatrix();
```

glrect

```
glrect(x1: number, y1: number, x2: number, y2: number);
```

Name	Type	Description
x1	number	
y1	number	
x2	number	
y2	number	

glrect

```
glrect(rectValues: number[]);
```

Name	Type	Description
rectValues	number[]	

glrotate

```
glrotate(angle: number, x: number, y: number, z: number);
```

Name	Type	Description
angle	number	
x	number	
y	number	
z	number	

glrotate

```
glrotate(rotateValues: number[]);
```

Name	Type	Description
rotateValues	number[]	

glscale

```
glscale(x: number, y: number, z: number);
```

Name	Type	Description
x	number	
y	number	
z	number	

glscale

```
glscale(scaleValues: number[]);
```

Name	Type	Description
scaleValues	number[]	

glscissor

```
glscissor(x: number, y: number, width: number, height: number);
```

Name	Type	Description
x	number	
y	number	
width	number	
height	number	

glscissor

```
glscissor(scissorValues: number[]);
```

Name	Type	Description
scissorValues	number[]	

glshademodel

```
glshademodel(mode: number);
```

Name	Type	Description
mode	number	

gltexcoord

```
gltexcoord(s: number, t: number);
```

Name	Type	Description
s	number	
t	number	

gltexenv

```
gltexenv(parameter_name: string, val1: number, val2: number, val3: number,
val4: number);
```

Name	Type	Description
parameter_name	string	
val1	number	
val2	number	
val3	number	
val4	number	

gltexgen

```
gltexgen(coord: number[], parameter_name: string, val1: number, val2:
number, val3: number, val4: number);
```

Name	Type	Description
coord	number[]	
parameter_name	string	
val1	number	
val2	number	
val3	number	
val4	number	

gltexparameter

```
gltexparameter(parameter_name: string, val1: number, val2: number, val3:
number, val4: number);
```

Name	Type	Description
parameter_name	string	
val1	number	
val2	number	
val3	number	
val4	number	

gltranslate

```
gltranslate(x: number, y: number, z: number);
```

Name	Type	Description
x	number	
y	number	
z	number	

gltranslate

```
gltranslate(translateValues: number[]);
```

Name	Type	Description
translateValues	number[]	

glulookat

```
glulookat(eye_x: number, eye_y: number, eye_z: number, center_x: number,
center_y: number, center_z: number, up_x: number, up_y: number, up_z:
number);
```

Name	Type	Description
eye_x	number	
eye_y	number	
eye_z	number	
center_x	number	
center_y	number	
center_z	number	
up_x	number	
up_y	number	
up_z	number	

glulookat

```
glulookat(lookatValues: number[]);
```

Name	Type	Description
lookatValues	number[]	

gluortho2d

```
gluortho2d(left: number, right: number, bottom: number, top: number);
```

Name	Type	Description
left	number	
right	number	
bottom	number	
top	number	

gluortho2d

```
gluortho2d(orthoValues: number[]);
```

Name	Type	Description
orthoValues	number[]	

gluperspective

```
gluperspective(fovy: number, aspect: number, near: number, far: number);
```

Name	Type	Description
fovy	number	
aspect	number	
near	number	
far	number	

gluperspective

```
gluperspective(perspectiveValues: number[]);
```

Name	Type	Description
perspectiveValues	number[]	

glvertex

```
glvertex(x: number, y: number, z: number);
```

Name	Type	Description
x	number	
y	number	
z	number	

glvertex

```
glvertex(vertexValues: number[]);
```

Name	Type	Description
vertexValues	number[]	

glviewport

```
glviewport(x: number, y: number, width: number, height: number);
```

Name	Type	Description
x	number	
y	number	
width	number	
height	number	

glviewport

```
glviewport(viewportValues: number[]);
```

Name	Type	Description
viewportValues	number[]	

line

Draw a line relative to the current position

Draws a line from the current drawing position to the location specified by adding the delta x, y, and z arguments to the current position. After this method has been called, the drawing position is updated by an offset relative to the original drawing position.

```
line(dx: number, dy: number, dz: number);
```

Name	Type	Description
dx	number	x offset
dy	number	y offset
dz	number	z offset

linesegment

Draw a line segment

Draws a line from the location specified by the x1, y1, and z1 arguments to the location specified by the x2, y2, and z2 arguments. After this method has been called, the drawing position is updated to the location specified by the x2, y2, and z2 arguments.

```
linesegment(x1: number, y1: number, z1: number, x2: number, y2: number,
z2: number);
```

Name	Type	Description
x1	number	x coordinate of the start point
y1	number	y coordinate of the start point
z1	number	z coordinate of the start point
x2	number	x coordinate of the end point
y2	number	y coordinate of the end point
z2	number	z coordinate of the end point

lineto

Draw a line to the specified position

Draws a line from the current drawing position to the location specified by the x, y, and z arguments. After this method has been called, the drawing position is updated to the location specified by the x, y, and z arguments.

```
lineto(x: number, y: number, z: number);
```

Name	Type	Description
x	number	x coordinate
y	number	y coordinate
z	number	z coordinate

move

Move the drawing position relatively

Moves the drawing position to the location specified by the sum of the current drawing position and the delta x, y, and z arguments.

```
move(dx: number, dy: number, dz: number);
```

Name	Type	Description
dx	number	x offset
dy	number	y offset
dz	number	z offset

moveto

Move the current drawing position

Moves the drawing position to the location specified by the x, y, and z arguments.

```
moveto(x: number, y: number, z: number);
```

Name	Type	Description
x	number	x coordinate
y	number	y coordinate
z	number	z coordinate

ortho3d

Set the default graphics state for rendering with orthographic projection

The orth3d method is a simple way to set the graphics state to default properties useful for 3D graphics, using an orthographic projection (i.e. object scale is not affected by distance from the camera). It is called every time the jsui object is resized if ortho3d() has been called more recently than default2d(), or default3d(). It is essentially equivalent to the following set of calls:

```
ortho3d();
```

Example

```
with (sketch) {
        glpolygonmode("front_and_back", "fill")
        glpointsize(1)
        gllinewidth(1)
        glenable("depth_test")
        glenable("fog")
        glcolor(0, 0, 0, 1)
        glshademodel("smooth")
        gllightmodel("two_side", "true")
        glenable("lighting")
        glenable("light0")
        glenable("normalize")
        gldisable("texture")
        glmatrixmode("projection")
        glloadidentity()
        glortho(-aspect, aspect, -1, 1, -1, 100)
        glmatrixmode("modelview")
        glloadidentity()
        glulookat(0, 0, 2, 0, 0, 0, 0, 0, 1)
        glclearcolor(1, 1, 1, 1)
        glclear()
        glenable("blend")
        glblendfunc("src_alpha", "one_minus_src_alpha")
}
```

plane

Draw a plane

Draws a plane with top width = 2 * scale_x1, left height = 2 * scale_y1, bottom width = 2 * scale_x2, right height = 2 * scale_y2, and center point at the current drawing position. If scale_y1 is not specified, it will assume the same value as scale_x1. If scale_x2 and scale_y2 are not specified, they will assume the same values as scale_x1 and scale_y1 respectively. Affected by shapeorient, shapeslice, and shapeprim values.

```
plane(scale_x1: number, scale_y1?: number, scale_x2?: number, scale_y2?:
number);
```

Name	Type	Description
scale_x1	number	half top width
optional scale_y1	number	half left height
optional scale_x2	number	half bottom width
optional scale_y2	number	half right height

point

Draw a point

Draws a point at the location specified by the x, y, and z arguments. After this method has been called, the drawing position is updated to the specified location.

```
point(x: number, y: number, z: number);
```

Name	Type	Description
x	number	x coordinate
y	number	y coordinate
z	number	z coordinate

quad

Draw a filled quadrilateral

After this method has been called, the drawing position is updated to the location specified by the x4, y4, and z4 arguments.

```
quad(x1: number, y1: number, z1: number, x2: number, y2: number, z2:
number, x3: number, y3: number, z3: number, x4: number, y4: number, z4:
number);
```

Name	Type	Description
x1	number	x coordinate of the first point
y1	number	y coordinate of the first point
z1	number	z coordinate of the first point
x2	number	x coordinate of the second point
y2	number	y coordinate of the second point
z2	number	z coordinate of the second point

x3	number	x coordinate of the third point
y3	number	y coordinate of the third point
z3	number	z coordinate of the third point
x4	number	x coordinate of the fourth point
y4	number	y coordinate of the fourth point
z4	number	z coordinate of the fourth point

roundedplane

Draw a plane with rounded corners

Draws a rounded plane with width = 2 * scale_x, height = 2 * scale_y, and center point at the current drawing position. The radius of the rounded portion of the plane is determined by the round_amount argument. If scale_y is not specified, it will assume the same value as scale_x. Affected by shapeorient, shapeslice, and shapeprim values.

```
roundedplane(round_amount: number, scale_x1: number, scale_y1?: number);
```

Name	Type	Description
round_amount	number	radius of the rounded corners
scale_x1	number	half width
optional scale_y1	number	half height

screentoworld

Return the world coordinate for a point on screen

Returns an array containing the x, y, and z world coordinates associated with a given screen pixel using the same the depth from the camera as 0, 0, 0. Optionally a third depth arg may be specified, which may be useful for hit detection and other applications. The depth value is typically specified in the range 0.-1. where 0 is the near clipping plane, and 1. is the far clipping plane. The worldtoscreen method can be used to determine the depth value of a given world coordinate, and the Sketch.depthatpixel() method can be used to determine the depth value associated with the currently rendered pixel at a given absolute screen coordinate.

```
screentoworld(x: number, y: number, depth?: number): [number, number,
number];
```

Name	Type	Description
x	number	screen x coordinate
y	number	screen y coordinate
optional depth	number	range from 0 (near clipping plane) to 1 (far clipping plane)
Return Value	[number, number, number]	

setpixel

Set the pixel value at a given location

Sets the pixel value at the specified location. Color values are floating point numbers in the range 0.-1.

```
setpixel(x: number, y: number, red: number, green: number, blue: number,
alpha: number);
```

Name	Type	Description
x	number	x
y	number	y
red	number	red
green	number	green
blue	number	blue
alpha	number	alpha

shapeorient

Set rotation for shape drawing calls

Sets the rotation in x, y, and z for future shape drawing calls.

```
shapeorient(rotation_x: number, rotation_y: number, rotation_z: number);
```

Name	Type	Description
rotation_x	number	x rotation in degrees
rotation_y	number	y rotation in degrees
rotation_z	number	z rotation in degrees

shapeprim

Set the drawing primitive shape

Sets the OpenGL drawing primitive to use within any of the "shape" drawing methods.

```
shapeprim(draw_prim: DrawingPrimitiveType);
```

Name	Type	Description
draw_prim	DrawingPrimitiveType	the drawing primitive to use

shapeslice

Set the number of slices to use when rendering shapes

Increasing the slice_a and slice_b arguments will increase the quality at which the shape is rendered, while decreasing these values will improve performance.

```
shapeslice(slice_a: number, slice_b: number);
```

Name	Type	Description
slice_a	number	number of slices to use
slice_b	number	number of slices to use

sphere

Draw a sphere

Draws a sphere with the given radius, centered at the current drawing position. If the theta1_start, theta1_end, theta2_start, and theta2_end arguments are specified, then a section will be drawn instead of a full sphere. Affected by shapeorient, shapeslice, and shapeprim values.

```
sphere(radius: number, theta1_start?: number, theta1_end?: number,
theta2_start?: number, theta2_end?: number);
```

Name	Type	Description
radius	number	radius of the sphere
optional theta1_start	number	start angle in degrees (0 - 360)
optional theta1_end	number	end angle in degrees (0 - 360)
optional theta2_start	number	start angle in degrees (0 - 360)
optional theta2_end	number	end angle in degrees (0 - 360)

strokeparam

Set the value for a given stroke param. Depending on the parameter, may apply to each point, or to the path as a whole. See Basic2dStrokeStyleParameterNames and LineStrokeStyleParameterNames.

```
strokeparam(parameter_name: Basic2dStrokeStyleParameterNames.alpha, value:
number);
```

Name	Type	Description
parameter_name	Basic2dStrokeStyleParameterNames.alpha	
value	number	

strokeparam

```
strokeparam(parameter_name: LineStrokeStyleParameterNames.order, order:
number = 3);
```

Name	Type	Description
parameter_name	LineStrokeStyleParameterNames.order	
optional order	number	

strokeparam

```
strokeparam(parameter_name: LineStrokeStyleParameterNames.slices,
slice_count: number = 20);
```

Name	Type	Description
parameter_name	LineStrokeStyleParameterNames.slices	
optional slice_count	number	

strokeparam

```
strokeparam(parameter_name: Basic2dStrokeStyleParameterNames.color, red:
number, green: number, blue: number, alpha: number);
```

Name	Type	Description
parameter_name	Basic2dStrokeStyleParameterNames.color	
red	number	

green	number
blue	number
alpha	number

strokeparam

```
strokeparam(parameter_name: Basic2dStrokeStyleParameterNames.order, order:
number = 3);
```

Name	Type	Description
parameter_name	Basic2dStrokeStyleParameterNames.order	
optional order	number	

strokeparam

```
strokeparam(parameter_name: Basic2dStrokeStyleParameterNames.outcolor,
red: number, green: number, blue: number, alpha: number);
```

Name	Type	Description
parameter_name	Basic2dStrokeStyleParameterNames.outcolor	
red	number	
green	number	
blue	number	
alpha	number	

strokeparam

```
strokeparam(parameter_name: Basic2dStrokeStyleParameterNames.outline,
active: 0 | 1);
```

Name	Type	Description
parameter_name	Basic2dStrokeStyleParameterNames.outline	
active	0 | 1	

strokeparam

```
strokeparam(parameter_name: Basic2dStrokeStyleParameterNames.scale, width:
number);
```

Name	Type	Description
parameter_name	Basic2dStrokeStyleParameterNames.scale	
width	number	

strokeparam

```
strokeparam(parameter_name: Basic2dStrokeStyleParameterNames.slices,
slice_count: number = 20);
```

Name	Type	Description
parameter_name	Basic2dStrokeStyleParameterNames.slices	
optional slice_count	number	

strokeparam

```
strokeparam(parameter_name: LineStrokeStyleParameterNames.alpha, value:
number);
```

Name	Type	Description
parameter_name	LineStrokeStyleParameterNames.alpha	
value	number	

strokeparam

```
strokeparam(parameter_name: LineStrokeStyleParameterNames.color, red:
number, green: number, blue: number, alpha: number);
```

Name	Type	Description
parameter_name	LineStrokeStyleParameterNames.color	
red	number	
green	number	
blue	number	
alpha	number	

strokepoint

Add an anchor point to the current path

Some stroke styles such as "basic2d" will ignore the z coordinate.

```
strokepoint(x: number, y: number, z: number);
```

Name	Type	Description
x	number	x
y	number	y
z	number	z

text

Draw the given text

Draws the text specified by the string argument at the current drawing position, taking into account the current font, fontsize, and text alignment. Text is strictly 2D, and does not take into account any world transformations. After calling the text method, if the x axis text alignment is set to "left", the current drawing position will be updated to reflect the world position associated with the end of the string. If the x axis text alignment is set to "right", the current drawing position will be updated to reflect the world position associated with the end of the string. If the x axis text alignment is set to "center", the current drawing position will remain unchanged.

```
text(text: string);
```

Name	Type	Description
text	string	text to draw

textalign

Set the text alignment in x and y

Sets the alignment of text to be drawn with respect to the current drawing position. Default alignment is "left" and "bottom".

```
textalign(align_x: "left" | "center" | "right", align_y: "top" | "center"
| "bottom");
```

Name	Type	Description
align_x	"left" | "center" | "right"	horizontal alignment
align_y	"top" | "center" | "bottom"	vertical alignment

torus

Draw a torus

Draw a torus centered at the current drawing position. If theta1_start, theta1_end, theta2_start, and theta2_end are specified, then a section will be drawn instead of a full torus. Affected by shapeorient, shapeslice, and shapeprim values.

```
torus(major_radius: number, minor_radius: number, theta1_start?: number,
theta1_end?: number, theta2_start?: number, theta2_end?: number);
```

Name	Type	Description
major_radius	number	major radius
minor_radius	number	minor radius
optional theta1_start	number	start angle in degrees (0 - 360)
optional theta1_end	number	end angle in degrees (0 - 360)
optional theta2_start	number	start angle in degrees (0 - 360)
optional theta2_end	number	end angle in degrees (0 - 360)

tri

Draw a filled triangle

After this method has been called, the drawing position is updated to the location specified by the x3, y3, and z3 arguments.

```
tri(x1: number, y1: number, z1: number, x2: number, y2: number, z2:
number, x3: number, y3: number, z3: number);
```

Name	Type	Description
x1	number	x coordinate of the first point
y1	number	y coordinate of the first point
z1	number	z coordinate of the first point
x2	number	x coordinate of the second point
y2	number	y coordinate of the second point
z2	number	z coordinate of the second point

y3	number	y coordinate of the third point
z3	number	z coordinate of the third point

worldtoscreen

Returns the screen coordinate for a given world coordinate

Returns an array containing the x, y, and depth screen coordinates associated with a given world coordinate. The depth value is typically specified in the range 0.-1. where 0 is the near clipping plane, and 1. is the far clipping plane.

```
worldtoscreen(x: number, y: number, z: number): [number, number, number];
```

Name	Type	Description
x	number	world x coordinate
y	number	world y coordinate
z	number	world z coordinate
Return Value	[number, number, number]	

class SnapshotAPI

Provides access to patcher snapshots.

SnapshotAPI is not yet supported in the v8 engine.

Constructors

```
new SnapshotAPI(varname: string);
```

Constructs a new instance of the `SnapshotAPI` class

Parameter	Type	Description
varname	string	the varname of an object to snapshot, or `'patcher'` for a patcher snapshot

Methods

addsnapshot

Create a snapshot at a given index, appending to the snapshot list if the index is already occupied.

```
addsnapshot(userpath?: string, index?: number, name?: string): void;
```

Name	Type	Description
optional userpath	string	the pathname to the maxsnap file within the user Snapshots directory (default: autogenerated)
optional index	number	the snapshot index (default: 0)
optional name	string	the snapshot name (default: name of object)

deletesnapshot

Delete a snapshot

```
deletesnapshot(index: number): void;
```

Name	Type	Description
index	number	the snapshot index

exportsnapshot

Save a snapshot to a file

```
exportsnapshot(index: number, userpath?: string): void;
```

Name	Type	Description
index	number	the snapshot index
optional userpath	string	the pathname to the maxsnap file within in the user Snapshots directory (default: autogenerated)

getembedsnapshot

Query the 'embed' state of a snapshot

```
getembedsnapshot(index: number): number;
```

Name	Type	Description
index	number	the snapshot index
Return Value	number	1 if the snapshot at the index is embedded, 0 if not

getnumsnapshots

Get the total number of snapshots

```
getnumsnapshots(): number;
```

Name	Type	Description
Return Value	number	

getsnapshotname

Get the name of the snapshot at a given index

```
getsnapshotname(index: number): string;
```

Name	Type	Description
index	number	the snapshot index
Return Value	string	

importsnapshot

Load a snapshot from a file into a given slot

```
importsnapshot(index?: number, userpath?: string): void;
```

Name	Type	Description
optional index	number	the snapshot index
optional userpath	string	the pathname to the maxsnap file within in the user Snapshots directory

movesnapshot

Change a snapshot's index

Does nothing if the `srcIndex` or `destIndex` doesn't exist

```
movesnapshot(srcIndex: number, destIndex: number): void;
```

Name	Type	Description
srcIndex	number	the current snapshot index
destIndex	number	the new snapshot index

restore

Restore a snapshot

```
restore(index: number): void;
```

Name	Type	Description
index	number	the snapshot index

setembedsnapshot

Set the embed state of a snapshot

```
setembedsnapshot(index: number, embedstate: number): void;
```

Name	Type	Description
index	number	the snapshot index
embedstate	number	1 if embedded, 0 if not

setsnapshotname

Set the name of the snapshot at the given index

```
setsnapshotname(index: number, name: string): void;
```

Name	Type	Description
index	number	the snapshot index
name	string	the snapshot name

snapshot

Create a snapshot at a given index

Unlike SnapshotAPI.addsnapshot(), `snapshot` will overwrite existing snapshots

```
snapshot(userpath?: string, index?: number, name?: string): void;
```

Name	Type	Description
optional userpath	string	the pathname to the maxsnap file within in the user Snapshots directory (default: autogenerated)
optional index	number	the snapshot index (default: 0)
optional name	string	the snapshot name (default: name of object)

class SQLite

Provides access to the SQLite database system.

A companion object, SQLResult, is required for most database operations.

SQLite is not yet supported in the v8 engine.

Constructors

```
new SQLite();
```

Constructs a new instance of the `SQLite` class

All future calls to the database will be through this instance of the object.

Methods

begintransaction

Start an SQL transaction on the database

This allows you to batch database updates and to roll back sets of changes if they do not all complete. When you are done with batch updates, a call to SQLite.endtransaction() should be executed.

```
begintransaction(): void;
```

close

Close a previously opened SQLite database

```
close(): void;
```

endtransaction

Complete a transaction and flush all database writes to the file

```
endtransaction(): void;
```

exec

Perform an SQL command on the database

This command must be in standard SQL language syntax, limited to the operations that SQLite supports.

```
exec(command: string, result: SQLResult): number;
```

Name	Type	Description
command	string	SQL command
result	SQLResult	SQLResult object to populate with transaction results
Return Value	number	an error code if unsuccessful or zero if the call results in a completed operation

Example

```
var res = new SQLResult;
var rtn = sqlite.exec("CREATE TABLE Persons (PersonID INTEGER, LastName
TEXT, FirstName TEXT);", res);
```

open

Open an SQLite-format file for database operations

```
open(filename: string, on_disk?: number, must_exist?: number): number;
```

Name	Type	Description
filename	string	File to access

optional must_exist	number	If non-zero, requires the file to exist to be opened, otherwise, a file will be created if one does not exist.
Return Value	number	an error code if unsuccessful or zero if the call results in an opened database

class SQLResult

A container for results obtained in an SQLite.exec() call.

Not every SQLite.exec() call will produce results, but any database query (`SELECT` in particular) will generate an SQLResult object even if the result is empty.

SQLResult is not yet supported in the v8 engine.

Constructors

```
new SQLResult();
```

Constructs a new instance of the `SQLResult` class

Methods

fieldname

Get the fieldname of a column at a given index

```
fieldname(index: number): string;
```

Name	Type	Description
index	number	column index
Return Value	string	the name of the column

numfields

Get the number of fields in the dataset returned in the SQLResult object

```
numfields(): number;
```

Name	Type	Description
Return Value	number	the number of fields

numrecords

Get the number of records were returned in the SQLResult object

```
numrecords(): number;
```

Name	Type	Description
Return Value	number	the number of records

value

Get the value of a record at a column index and record number

```
value(index: number, record_no: number): number | string;
```

Name	Type	Description
index	number	column index
record_no	number	record number
Return Value	number | string	the value of the record

Example

```
function print_everything(sqlres) {
    var numrecs = sqlres.numrecords()
    var numflds = sqlres.numfields()

    var field_names = new Array()
    for (var i = 0; i < numflds; i++) {
        field_names[i] = sqlres.fieldname(i)
    }

    for (var i = 0; i < numrecs; i++) {
        for (var j = 0; j < numflds; j++) {
            post(
                "Rec: ",
                i,
                " field ",
                field_names[j],
                " value ",
                sqlres.value(j, i),
                "\n"
            )
        }
    }
}
```

class Task

A function that can be scheduled or repeated.

Although the overall timing accuracy of a Task function is high, the latency between the scheduled time and the actual execution time of a Task function is variable because the function runs in a low-priority thread. Therefore, you should avoid using a Task function in a time-critical operation.

For convenience, a Task object is a property of the function executed in a Task. To access the Task from within its function, use the following syntax:

```
arguments.callee.task
```

Example

```
function ticker(a, b, c) {
    post("tick")
}

args = new Array(3)
args[0] = 1
args[1] = 2
args[2] = 3
t = new Task(ticker, this, args)
```

Constructors

```
new Task(fn: Function, obj?: object, args?: any[]);
```

Constructs a new instance of the `Task` class

Parameter	Type	Description
fn	Function	the function to execute
optional obj	object	the `this` during the execution of the function
optional args	any[]	the arguments to pass to `fn`

Properties

arguments any[]

The arguments passed to the task's executing function

function Function

The function executed in the Task. You can change this within the task function itself.

interval number

The time in milliseconds between repeats of the task function. The default interval is 500 ms.

Example

This example will cause the task to run 10% slower each time the function is called.

```
function taskfun() {
    var intv = arguments.callee.task.interval
    arguments.callee.task.interval = intv + intv * 0.1
}
```

iterations number

read-only

The number of times the task function has been called.

Outside of a task function, the value of iteration is always 0. The value resets each time the task is started (using the Task.repeat(), Task.execute(), or Task.schedule() methods).

object object

The object assigned to be the `this` in the task function. Most often this will be your `jsthis` object, so you can, for example, access the `outlet()` method. You set up your `jsthis` object to be the `this` by creating a task with the keyword `this` as the first argument.

Example

```
arguments.callee.task.object.outlet(1, "bang")
outlet(1, "bang")
this.outlet(1, "bang")
```

running boolean

read-only

Whether the Task is running or not. Within a function executing within a task, this will always be true.

valid boolean

Whether the Task object has been invalidated and is awaiting garbage collection. An invalid object will no longer respond to the Task.execute() or Task.schedule() methods.

Methods

cancel

If a task is scheduled or repeating, cancel any future executions. This method can be used within a task function for a self-cancelling Task.

```
cancel(): void;
```

Example

The following is a task function that will run only once, even if it is started using the Task.repeat() function.

```
function once() {
    arguments.callee.task.cancel()
}
```

execute

Run the task once, right now. Equivalent to calling the task function with its arguments.

```
execute(): void;
```

freepeer

Invalidate the Task and make it available for garbage collection by the JS engine.

Task objects persist beyond their code scope (otherwise, the object could be garbage collected before its scheduled function is called). The user is responsible for invalidating the Task when it is no longer in use. All Tasks (valid or invalid) will be garbage collected and freed when the parent JS object reloads its script or is itself freed.

```
freepeer(): void;
```

Example

```
function bang() {
    var tsk = new Task(cb) // Task will not be freed when the bang()
function returns
    tsk.schedule(200)
}

function cb() {
    post("right on schedule!\n")
    arguments.callee.task.freepeer() // ensure that the caller can be GC'd
}
```

repeat

Repeat a task function

```
repeat(n?: number, initialdelay?: number): void;
```

Name	Type	Description
optional n	number	Number of repetitions. If none is provided, the task repeats until cancelled.
optional initialdelay	number	Delay in milliseconds until the first iteration

Example

```
function repeater_function() {
    post(arguments.callee.task.iterations)
}

tsk = new Task(repeater_function, this)
tsk.interval = 1000 // every second
tsk.repeat(3) // do it 3 times
```

schedule

Run the task once, with a delay.

```
schedule(delay?: number): void;
```

Name	Type	Description
optional delay	number	Time in milliseconds before the task function will be executed

class Wind

A property of the Patcher which represents its window.

You cannot create a new Wind or access other types of windows (such as that of a Max table object).

Properties

assoc Patcher

read-only

The Patcher object associated with the window.

assocclass string

read-only

The Max class of the object associated with the window.

dirty boolean

Whether the window's contents have been modified.

This property is read-only in the runtime version of Max.

hasgrow boolean

Whether the window has a grow area.

hashorizscroll boolean

read-only

Whether the window has a horizontal scroll bar.

hastitlebar boolean

Whether the window has a title bar.

hasvertscroll boolean

read-only

Whether the window has a vertical scroll bar.

haszoom boolean

Whether the window has a zoom box.

location [number, number, number, number]

An array of the four coordinates (left, top, right, bottom) representing the window's location in global coordinates.

next Wind

read-only

The Wind object of the next patcher visible in the application's list of windows. The first Wind object can be accessed using the Max.frontpatcher `wind` property.

size [number, number]

An array of two coordinates (width, height) representing the window's size

title string

The window's title

visible boolean

Whether the window is visible

Methods

bringtofront

Move the window in front of all other windows.

```
bringtofront(): void;
```

scrollto

Scroll the window.

```
scrollto(x: number, y: number): void;
```

Name	Type	Description
x	number	The x-coordinate of the new top-left corner
y	number	The y-coordinate of the new top-left corner

sendtoback

Move the window behind all other windows

```
sendtoback(): void;
```

setlocation

Set the global location of the window.

```
setlocation(left: number, top: number, bottom: number, right: number):
void;
```

Name	Type	Description
left	number	x-coordinate of the new top-left corner
top	number	y-coordinate of the new top-left corner
bottom	number	y-coordinate of the new bottom-right corner
right	number	x-coordinate of the new bottom-right corner

Credits

Cycling '74
440 N Barranca Ave #4074
Covina, CA 91723 USA

Contact Support: https://cycling74.com/support/contact