Reference:GeoGebra Apps API

(Redirected from Reference:JavaScript)

Jump to: navigation, search

This page describes the GeoGebra Apps API to interact with GeoGebra apps. Please see GeoGebra Apps Embedding on how to embed our apps into your web pages.

Contents

[hide]

Examples

In these examples you can see the GeoGebra Apps API in action:

- Showing & hiding objects with buttons
- Saving & loading state
- Listening to update, add, remove events

Note: Arguments in square brackets can be omitted.

Commands and Undo-Points

Method Signature	Since	Description
boolean evalCommand(String cmdString)	3.0	Evaluates the given string just like it would be evaluated when entered into GeoGebra's input bar. Returns whether command evaluation was successful. From GeoGebra 3.2 you can pass multiple commands at once by separating them with \n. Note: you must use English commands names
String[] evalCommandGetLabels(String cmdString)	5.0	Like evalCommand(), but the return value is a String containing a comma-separated list of the labels of the created objects eg `"A, B, C"`
String evalCommandCAS(String string)	3.2	Passes the string to GeoGebra's CAS and returns the result as a String.
void setUndoPoint()	3.2	Sets an undo point. Useful if you want the user to be able to undo that action of evalCommand eg if you have made an HTML button to act as a custom tool

Setting the state of objects

General methods

Method Signature	Since	Description
void deleteObject(String objName)	2.7	Deletes the object with the given name.
void setAuxiliary(geo, true/false)	5.0	Affects or not the status of "auxiliary object" to object *geo*.
void setValue(String objName, double value)	3.2	Sets the double value of the object with the given name. Note: if the specified object is boolean, use a value of 1 to set it to true and any other value to set it to false. For any other object type, nothing is done.
void setTextValue(String objName, String value)	3.2	Sets the text value of the object with the given name. For any other object type, nothing is done.

void setListValue(String objName, int i, double value)	5.0	Sets the value of the list element at position 'i' to 'value'
void setCoords(String objName, double x, double y) void setCoords(String objName, double x, double y, double z)	3.0 5.0	Sets the coordinates of the object with the given name. Note: if the specified object is not a point, vector, line or absolutely positioned object (text, button, checkbox, input box) nothing is done.
void setCaption(String objName, String caption)	5.0	Sets the caption of object with given name.
void setColor(String objName, int red, int green, int blue)	2.7	Sets the color of the object with the given name.
void setVisible(String objName, boolean visible)	2.7	Shows or hides the object with the given name in the graphics window.
void setLabelVisible(String objName, boolean visible)	3.0	Shows or hides the label of the object with the given name in the graphics window.
void setLabelStyle(String objName, int style)	3.0	Sets the label style of the object with the given name in the graphics window. Possible label styles are NAME = 0, NAME_VALUE = 1, VALUE = 2 and (from GeoGebra 3.2) CAPTION = 3
void setFixed(String objName, boolean fixed, boolean selectionAllowed)	3.0	Sets the "Fixed" and "Selection Allowed" state of the object with the given name. Note: fixed objects cannot be changed.
void setTrace(String objName, boolean flag)	3.0	Turns the trace of the object with the given name on or off.
boolean renameObject(String oldObjName, String newObjName)	3.2	Renames oldObjName to newObjName. Returns whether the rename was successful

void setLayer(String objName, int layer)	3.2	Sets the layer of the object
void setLayerVisible(int layer, boolean visible)	3.2	Shows or hides the all objects in the given layer
void setLineStyle(String objName, int style)	3.2	Sets the line style for the object (0 to 4)
void setLineThickness(String objName, int thickness)	3.2	sets the thickness of the object (1 to 13, -1 for default)
void setPointStyle(String objName, int style)	3.2	Sets the style of points (-1 default, 0 filled circle, 1 circle, 2 cross, 3 plus, 4 filled diamond, 5 unfilled diamond, 6 triangle (north), 7 triangle (south), 8 triangle (east), 9 triangle (west))
void setPointSize(String objName, int size)	3.2	Sets the size of a point (from 1 to 9)
void setDisplayStyle(String objName, String style)	5.0	Sets the display style of an object. Style should be one of "parametric", "explicit", "implicit", "specific"
void setFilling(String objName, double filling)	3.2	Sets the filling of an object (from 0 to 1)
String getPNGBase64(double exportScale, boolean transparent, double DPI)	4.0	Returns the active Graphics View as a base64-encoded String eg var str = ggbApplet.getPNGBase64(1, true, 72); The DPI setting is slow, set to `undefined` if you don't need it
void getScreenshotBase64(function callback)	5.0	Gets the screenshot of the whole applet as PNG and sends it to the callback function as a base64 encoded string. Example: `ggbApplet.getScreenshotBase64(function(url){window.open(""+url);});` **For internal use only, may not work in all browsers**
boolean writePNGtoFile(String filename, double exportScale,	4.0	Exports the active Graphics View to a .PNG file. The DPI setting is slow, set to `undefined` if you don't need it eg var success = ggbApplet.writePNGtoFile("myImage.png", 1, false, 72);

boolean transparent, double DPI)		
boolean isIndependent(String objName)	4.0	checks if **objName** is independent
boolean isMoveable(String objName)	4.0	checks if **objName** is is moveable
String getBase64()		Gets the current construction as a base64-encoded .ggb file
String getBase64(function callback)	4.2	Gets the current construction as a base64-encoded .ggb file asynchronously, passes as parameter to the callback function when ready. The callback function should take one parameter (the base64 string).
void setBase64(String [, function callback])	4.0	Sets the current construction from a base64-encoded .ggb file. If callback function is specified, it is called after the file is loaded.

Automatic Animation

Method Signature	Since	Description
void setAnimating(String objName, boolean animate)	3.2	Sets whether an object should be animated. This does not start the animation yet, use startAnimation() to do so.
void setAnimationSpeed(String objName, double speed)	3.2	Sets the animation speed of an object.
void startAnimation()	3.2	Starts automatic animation for all objects with the animating flag set, see setAnimating()
void stopAnimation()	3.2	Stops animation for all objects with the animating flag set, see setAnimating()
boolean isAnimationRunning()	3.2	Returns whether automatic animation is currently running.

Getting the state of objects

Method Signature	Since	Description
double getXcoord(String objName)	2.7	Returns the cartesian x-coord of the object with the given name. Note: returns 0 if the object is not a point or a vector.
double getYcoord(String objName)	2.7	Returns the cartesian y-coord of the object with the given name. Note: returns 0 if the object is not a point or a vector.
double getZcoord(String objName)	5.0	Returns the cartesian z-coord of the object with the given name. Note: returns 0 if the object is not a point or a vector.
double getValue(String objName)	3.2	Returns the double value of the object with the given name (e.g. length of segment, area of polygon). Note: returns 1 for a boolean object with value true. Otherwise 0 is returned.
double getListValue(String objName, Integer index)	5.0	Returns the double value of the object in the list (with the given name) with the given index. Note: returns 1 for a boolean object with value true. Otherwise 0 is returned.
String getColor(String objName)	2.7	Returns the color of the object with the given name as a hex string, e.g. "#FF0000" for red. Note that the hex string always starts with # and contains no lower case letters.
boolean getVisible(String objName)	3.2	Returns true or false depending on whether the object is visible in the Graphics View. Returns false if the object does not exist.
boolean getVisible(String objName, int view)	4.2	Returns true or false depending on whether the object is visible in Graphics View 'view' (1 or 2). Returns false if the object does not exist.
String getValueString(String objName)	2.7	Returns the value of the object with the given name as a string.
String getDefinitionString(String objName[, boolean useLocalizedInput])	2.7	Returns the description of the object with the given name as a string. If useLocalizedInput is false, returns the description in English, otherwise in current GUI language. Note: Localized input uses parentheses, non-localized input uses brackets.

String getCommandString(String objName [, boolean useLocalizedInput])	5.0	Returns the command of the object with the given name as a string. If useLocalizedInput is false, returns the command in English, otherwise in current GUI language. Note: Localized input uses parentheses, non-localized input uses brackets.
String getLaTeXString(String objName)	5.0	Returns the value of given object in LaTeX syntax
String getLaTeXBase64(String objName, boolean value)	5.0	Returns base64 encoded PNG picture containing the object as LaTeX. For value = false the object is represented as the definition, for value=true the object value is used.
String getObjectType(String objName)	2.7	Returns the type of the given object as a string (like "point", "line", "circle", etc.).
boolean exists(String objName)	2.7	Returns whether an object with the given name exists in the construction.
boolean isDefined(String objName)	2.7	Returns whether the given object's value is valid at the moment.
String [] getAllObjectNames([String type]) ***Deprecated since 3.0***	2.7	Returns an array with all object names in the construction. If type parameter is entered, only objects of given type are returned.
int getObjectNumber()	3.0	Returns the number of objects in the construction.
int getCASObjectNumber()	3.0	Returns the number of object (nonempty cells) in CAS.
String getObjectName(int i)	3.0	Returns the name of the n-th object of the construction.
String getLayer(String objName)	3.2	Returns the layer of the object.
int getLineStyle(String objName)	3.2	Gets the line style for the object (0 to 4)
int getLineThickness(String objName)	3.2	Gets the thickness of the line (1 to 13)
int getPointStyle(String objName)	3.2	Gets the style of points (-1 default, 0 filled circle, 1 circle, 2 cross, 3 plus, 4 filled diamond, 5 unfilled diamond, 6 triangle (north), 7 triangle (south), 8 triangle (east), 9 triangle (west))
int getPointSize(String objName)	3.2	Gets the size of a point (from 1 to 9)
double getFilling(String objName)	3.2	Gets the filling of an object (from 0 to 1)

getCaption(String objectName, boolean substitutePlaceholders)	5.0	Returns the caption of the object. If the caption contains placeholders (%n, %v,...), you can use the second parameter to specify whether you want to substitute them or not.
getLabelStyle(String objectName)	5.0	Returns label type for given object, see setLabelStyle for possible values.
getLabelVisible()	5.0	

Construction / User Interface

Method Signature	Since	Description
void setMode(int mode)	2.7	Sets the mouse mode (i.e. tool) for the graphics window (see toolbar reference and the applet parameters "showToolBar" and "customToolBar")
int getMode()	5.0	Gets the mouse mode (i.e. tool), see toolbar reference for details
void openFile(String strURL)	2.7 (Java only)	Opens a construction from a file (given as absolute or relative URL string)
void reset()	2.7	Reloads the initial construction (given in filename parameter) of this applet.
void newConstruction()	2.7	Removes all construction objects
void refreshViews()	2.7	Refreshs all views. Note: this clears all traces in the graphics window.
void setOnTheFlyPointCreationActive(boolean flag)	3.2	Turns on the fly creation of points in graphics view on (true) or off (false). Note: this is useful if you don't want tools to have the side effect of creating points. For example, when this flag is set to false, the tool "line through two points" will not create points on the fly when you click on the background of the graphics view.
void setPointCapture(view, mode)	5.0	Change point capturing mode.

		view: 1 for graphics, 2 for graphics 2, -1 for 3D. mode: 0 for no capturing, 1 for snap to grid, 2 for fixed to grid, 3 for automatic.
void setRounding(string round)	5.0	The string consists of a number and flags, "s" flag for significant digits, "d" for decimal places (default). JavaScript integers are cast to string automaticlly. Example: "10s", "5", 3
void hideCursorWhenDragging(boolean flag)	3.2	Hides (true) or shows (false) the mouse cursor (pointer) when dragging an object to change the construction.
void setRepaintingActive(boolean flag)	2.7	Turns the repainting of this applet on (true) or off (false). Note: use this method for efficient repainting when you invoke several methods.
void setErrorDialogsActive(boolean flag)	3.0	Turns showing of error dialogs on (true) or off (false). Note: this is especially useful together with evalCommand().
void setCoordSystem(double xmin, double xmax, double ymin, double ymax)	3.0	Sets the Cartesian coordinate system of the graphics window.
void setCoordSystem(double xmin, double xmax, double ymin, double ymax, double zmin, double zmax, boolean yVertical)	5.0	Sets the Cartesian coordinate system of the 3D graphics window. The last parameter determines whether y-axis should be oriented vertically.
void setAxesVisible(boolean xAxis, boolean yAxis)	3.0	Shows or hides the x- and y-axis of the coordinate system in the graphics windows 1 and 2.
void setAxesVisible(int viewNumber, boolean xAxis, boolean yAxis, boolean zAxis)	5.0	Shows or hides the x-, y- and z-axis of the coordinate system in given graphics window. **Example:** `ggbApplet.setAxesVisible(3, false, true, true)`
void setAxisLabels(int viewNumber, boolean xAxis, boolean yAxis, boolean zAxis)	5.0	Set label for the x-, y- and z-axis of the coordinate system in given graphics window.

		Example: `ggbApplet.setAxisLabels(3,"larg","long","area")`
void setAxisSteps(int viewNumber, boolean xAxis, boolean yAxis, boolean zAxis)	5.0	Set distance for the x-, y- and z-axis of the coordinate system in given graphics window. **Example:** `ggbApplet.setAxisSteps(3, 2,1,0.5)`
void setAxisUnits(int viewNumber, boolean xAxis, boolean yAxis, boolean zAxis)	5.0	Set units for the x-, y- and z-axis of the coordinate system in given graphics window. **Example:** `ggbApplet.setAxisUnits(3, "cm","cm","cm²")`
void setGridVisible(boolean flag)	3.0	Shows or hides the coordinate grid in the graphics windows 1 and 2.
void setGridVisible(int viewNumber, boolean flag)	5.0	Shows or hides the coordinate grid in given graphics view graphics window.
getGridVisible(int viewNumber)	5.0	Returns true if grid is visible in given view. If view number is omited, returns whether grid is visible in the first graphics view.
getPerspectiveXML()	5.0	Returns an XML representation of the current perspective.
undo()	5.0	Undoes one user action.
redo()	5.0	Redoes one user action.
showToolBar(boolean show)	HTML5	Sets visibility of toolbar
setCustomToolBar(String toolbar)	5.0	Sets the layout of the main toolbar, see toolbar reference for details
showMenuBar(boolean show)	HTML5	Sets visibility of menu bar
showAlgebraInput(boolean show)	HTML5	Sets visibility of input bar
showResetIcon(boolean show)	HTML5	Sets visibility of reset icon
enableRightClick(boolean enable)	5.0	Enables or disables right click features
enableLabelDrags(boolean enable)	5.0	Enables or disables dragging object labels
enableShiftDragZoom(boolean enable)	5.0	Enables or disables zooming and dragging the view using mouse or keyboard
enableCAS(boolean enable)	5.0	Enables or disables CAS features (both the view and commands)

enable3D(boolean enable)	5.0	Enables or disables the 3D view
void setPerspective(string perspective)	5.0	Changes the open views, see SetPerspective Command for the string interpretation.
setWidth(int width)	5.0 (HTML5)	Change width of the applet (in pixels)
setHeight(int height)	5.0 (HTML5)	Change height of the applet (in pixels)
setSize(int width, int height)	5.0 (HTML5)	Change width and height of the applet (in pixels)
recalculateEnvironments()	5.0 (HTML5)	Update the applet after scaling by external CSS

Event listeners

With these methods you can implement Applet to JavaScript communication. For example, these methods can be used to:

- monitor user actions (see Event listeners example)
- communicate between two GeoGebra applets (see two applets example)

Method Signature	Since	Description
void registerAddListener(String JSFunctionName)	3.0	Registers a JavaScript function as an **add** listener for the applet's construction. Whenever a new object is created in the GeoGebraApplet's construction, the JavaScript function *JSFunctionName* is called using the name of the newly created object as its single argument.

		Example: First, register a listening JavaScript function:
		
		`ggbApplet.registerAddListener("myAddListenerFunction");`
		When an object "A" is created, the GeoGebra Applet will call the Javascript function
		
		`myAddListenerFunction("A");`

void unregisterAddListener(String objName)	3.0	Removes a previously registered add listener, see *registerAddListener()*

void registerRemoveListener(String JSFunctionName)	3.0	Registers a JavaScript function as a **remove** listener for the applet's construction. Whenever an object is deleted from the GeoGebraApplet's construction, the JavaScript function *JSFunctionName* is called using the name of the deleted object as its single argument. Note: when a construction is cleared, remove is not called for every single object, see registerClearListener().
		Example: First, register a listening JavaScript function:
		
		`ggbApplet.registerRemoveListener("myRemoveListenerFunction");`
		When the object "A" is deleted, the GeoGebra Applet will call the Javascript function
		
		`myRemoveListenerFunction("A");`

void unregisterRemoveListener(String objName)	3.0	Removes a previously registered remove listener, see *registerRemoveListener()*
void registerUpdateListener(String JSFunctionName)	3.0	Registers a JavaScript function as a **update** listener for the applet's construction. Whenever any object is updated in the GeoGebraApplet's construction, the JavaScript function *JSFunctionName* is called using the name of the updated object as its single

		argument. Note: when you only want to listen for the updates of a single object use registerObjectUpdateListener() instead.
		Example: First, register a listening JavaScript function:
		
		`ggbApplet.registerUpdateListener("myUpdateListenerFunction");`
		When the object "A" is updated, the GeoGebra Applet will call the Javascript function
		
		`myUpdateListenerFunction("A");`

void unregisterUpdateListener(String objName)	3.0	Removes a previously registered update listener, see *registerUpdateListener()*
void registerClickListener(String JSFunctionName)	5.0	Registers a JavaScript function as a **click** listener for the applet's construction. Whenever any object is clicked in the GeoGebraApplet's construction, the JavaScript function *JSFunctionName* is called using the name of the updated object as its single argument. Note: when you only want to listen for the updates of a single object use registerObjectClickListener() instead. >
void unregisterClickListener(String objName)	3.0	Removes a previously registered click listener, see *registerClickListener()*
void registerObjectUpdateListener(String objName, String JSFunctionName)	3.0	Registers a JavaScript function as an **update** listener for a single object. Whenever the object with the given name is updated, the JavaScript function *JSFunctionName*is called using the name of the updated object as its single argument. If objName previously had a mapping JavaScript function, the old value is replaced. Note: all object updated listeners are unregistered when their object is removed or the construction is cleared, see registerRemoveListener() and registerClearListener().

		Example: First, register a listening JavaScript function:
		`ggbApplet.registerObjectUpdateListener("A", "myAupdateListenerFunction");`
		Whenever the object A is updated, the GeoGebra Applet will call the Javascript function
		`myAupdateListenerFunction("A");`
		Note: an object update listener will still work after an object is renamed.

void unregisterObjectUpdateListener(String objName)	3.0	Removes a previously registered object update listener of the object with the given name, see *registerObjectUpdateListener()*

void registerObjectClickListener(String objName, String JSFunctionName)	5.0	Registers a JavaScript function as an **click** listener for a single object. Whenever the object with the given name is clicked, the JavaScript function *JSFunctionName*is called using the name of the updated object as its single argument. If objName previously had a mapping JavaScript function, the old value is replaced. Note: all object click listeners are unregistered when their object is removed or the construction is cleared, see registerRemoveListener() and registerClearListener().
		Example: First, register a listening JavaScript function:
		`ggbApplet.registerObjectUpdateListener("A", "myAupdateListenerFunction");`
		Whenever the object A is clicked, the GeoGebra Applet will call the Javascript function
		`myAupdateListenerFunction("A");`

		Note: an object update listener will still work after an object is renamed.
void unregisterObjectClickListener(String objName)	5.0	Removes a previously registered object click listener of the object with the given name, see *registerObjectClickListener()*

void registerRenameListener(String JSFunctionName)	3.0	Registers a JavaScript function as a **rename** listener for the applet's construction. Whenever an object is renamed in the GeoGebraApplet's construction, the JavaScript function *JSFunctionName* is called using the old name and the new name of the renamed object as its two arguments.
		Example: First, register a listening JavaScript function:
		`ggbApplet.registerRenameListener("myRenameListenerFunction");`
		When an object "A" is renamed to "B", the GeoGebra Applet will call the Javascript function
		`myRenameListenerFunction("A", "B");`

void unregisterRenameListener(String objName)	3.0	Removes a previously registered rename listener, see *registerRenameListener()*
void registerClearListener(String JSFunctionName)	3.0	Registers a JavaScript function as a **clear** listener for the applet's construction. Whenever the construction in the GeoGebraApplet is cleared (i.e. all objects are removed), the JavaScript function JSFunctionName is called using no arguments. Note: all update listeners are unregistered when a construction is cleared. See registerUpdateListener() and registerRemoveListener().

		Example: First, register a listening JavaScript function:
		
		`ggbApplet.registerClearListener("myClearListenerFunction");`
		When the construction is cleared (i.e. after reseting a construction or opening a new construction file), the GeoGebra Applet will call the Javascript function
		
		`myClearListenerFunction();`

void unregisterClearListener(String JSFunctionName)	3.0	Removes a previously registered clear listener, see *registerClearListener()*
void registerStoreUndoListener(String JSFunctionName)	4.4	Registers a listener that is called (with no arguments) every time an undo point is created.
void unregisterStoreUndoListener(String JSFunctionName)	4.4	Removes previously registered listener for storing undo points, see registerStoreUndoListener

GeoGebra's XML format

With these methods you can set everything in a construction (see XML Reference).

Method Signature	Since	Description

void evalXML(String xmlString)	2.7	Evaluates the given XML string and changes the current construction. Note: the construction is NOT cleared before evaluating the XML string.
void setXML(String xmlString)	2.7	Evaluates the given XML string and changes the current construction. Note: the construction is cleared before evaluating the XML string. This method could be used to load constructions.
String getXML()	2.7	Returns the current construction in GeoGebra's XML format. This method could be used to save constructions.
String getXML(String objName)	3.2	Returns the GeoGebra XML string for the given object, i.e. only the <element> tag is returned.
String getAlgorithmXML(String objName)	3.2	For a dependent GeoElement objName the XML string of the parent algorithm and all its output objects is returned. For a free GeoElement objName "" is returned.

Miscellaneous

Method Signature	Since	Description
void debug(String string)	3.2	Prints the string to the Java Console
String getVersion()	5.0	Returns the version of GeoGebra

Automatic Exercise Checking (beta)

Method Signature	Since	Description
JSONObject getExerciseResult()	5.0	If there are Macros or an Exercise present in the current file this can be used to check if parts of the construction are equivalent to the Macros in the file. If you don't want that a Standard Exercise (using all the Macros in the Construction and setting each fraction to 100) will be created, check if this is a Exercise with isExercise() first. Hint will be empty unless specified otherwise with the ExerciseBuilder.

		Fraction will be 0 or 1 unless specified otherwise with the ExerciseBuilder. Result will be in Result,i.e: CORRECT, The assignment is CORRECT WRONG, if the assignment is WRONG and we can't tell why NOT_ENOUGH_INPUTS if there are not enough input geos, so we cannot check WRONG_INPUT_TYPES, if there are enough input geos, but one or more are of the wrong type WRONG_OUTPUT_TYPE, if there is no output geo matching our macro WRONG_AFTER_RANDOMIZE, if the assignment was correct in the first place but wrong after randomization UNKNOWN, if the assignment could not be checked
float getExerciseFraction()	5.0	If there are Macros or an Exercise present in the current file this can be used to check if parts of the construction are equivalent to the Macros in the file. It will return the overall Fraction of the Exercise. This is the sum of all the Fractions in the Assignment or 1 if one of the Assignments has a fraction of 100 and no negative fractions are present. Use getExerciseResult() to get the fractions of each Assignment. If you don't want that a standard exercise (using all the Macros in the Construction and setting each fraction to 100) will be created, check if this is a Exercise with isExercise() first.
boolean isExercise()	5.0	Check whether this applet is an Exercise Returns:true if the Exercise has assignments, this will happen when either getExerciseResult() or getExerciseFraction() are called with user defined Tools present in the applet or if the ExerciseBuilderDialog was used to create the Exercise.
boolean startExercise()	5.0	If you want to make use of the values of random geos a BoolAssignment depends on, this is an easy way to retrieve these values for displaying in the question text and stop randomizing them in order to store the same assignment that was presented to the student.

An example using this API functions can be found here:

```
http://dev.geogebra.org/examples/html/example10.html
```

Obtaining the Applet Object

In the basic cases the applet is in global *ggbApplet* variable. For multiple applets on one page, by default only one is accessible from JavaScript. To access multiple applets, you should set the "id" parameter of particular applets, it will become the name of global variable representing the applet

es:Referencia:JavaScript fr:Référence:JavaScript it:Riferimenti:JavaScript