



JavaScript Reference

Version: 1.3

Contents

1. Introduction.....	6
2. About ShortCAD JavaScript	7
3. Getting Started.....	8
4. ShortCAD Drawing Object model	10
4.1. WPoint Object	10
WPoint Object Properties.....	10
Constructor of WPoint Object	11
WPoint.equals Method	11
WPoint.distance Method	11
WPoint.angle Method	12
WPoint.rotate Method	12
WPoint.clone Method	12
4.2. DwgLAYER Object	13
DwgLAYER Object Properties.....	13
4.3. DwgLINETYPE Object	14
DwgLINETYPE Object Properties	14
4.4. DwgTEXTSTYLE Object.....	15
DwgTEXTSTYLE Object Properties	15
4.5. DwgObject Object.....	16
DwgObject Object Properties.....	16
DwgObject.draw Method.....	17
DwgObject.move Method	18
DwgObject.rotate Method	18
DwgObject.scale Method	18
DwgObject.erase Method	19
DwgObject.forget Method	19
DwgObject.createCopy Method.....	19
DwgObject.intersections Method	20
DwgObject.sameAs Method.....	20

4.6. DwgPOINT Object	21
DwgPOINT Object Properties	21
DwgPOINT Object Methods.....	21
4.7. DwgLINE Object	22
DwgLINE Object Properties	22
DwgLINE Object Methods.....	23
4.8. DwgSOLID Object.....	24
DwgSOLID Object Properties	24
DwgSOLID Object Methods	26
4.9. DwgTRACE Object.....	27
DwgTRACE Object Properties	27
DwgTRACE Object Methods	29
4.10. DwgARC Object.....	30
DwgARC Object Properties	30
DwgARC Object Methods	31
4.11. DwgCIRCLE Object	32
DwgCIRCLE Object Properties	32
DwgCIRCLE Object Methods.....	32
4.12. DwgELLIPSE Object	34
DwgELLIPSE Object Properties	34
DwgELLIPSE Object Methods.....	35
4.13. DwgPOLYLINE Object.....	37
DwgPOLYLINE Object Properties	37
DwgPOLYLINE Object Methods	37
DwgPolyline.getVertexes Method.....	38
DwgPolyline.appendVertex Method	38
4.14. DwgVERTEX Object.....	41
DwgVERTEX Object Properties	41
DwgVERTEX Object Methods	42
4.15. DwgTEXT Object	43

DwgTEXT Object Properties.....	43
DwgTEXT Object Methods.....	44
4.16. DwgINSERTION Object.....	45
DwgINSERTION Object Properties.....	45
DwgINSERTION Object Methods.....	46
4.17. drawing Variable.....	47
drawing Variable Properties.....	47
drawing.createPoint Method.....	51
drawing.createLine Method.....	53
drawing.createPolyline Method.....	55
drawing.createSolid Method.....	57
drawing.createTrace Method.....	59
drawing.createArc Method.....	62
drawing.createCircle Method.....	65
drawing.createEllipse Method.....	68
drawing.createText Method.....	71
drawing.redraw Method.....	74
drawing.getEntities Method.....	74
drawing.getSelectedEntities Method.....	75
drawing.setSelectedEntities Method.....	75
drawing.getLayers Method.....	76
drawing.getLayer Method.....	76
drawing.createLayer Method.....	77
drawing.getLineTypes Method.....	78
drawing.getLineType Method.....	78
drawing.getTextStyles Method.....	79
drawing.getTextStyle Method.....	79
drawing.selectObject Method.....	80
drawing.selectObjects Method.....	81
drawing.getPoint Method.....	83

- 5. ShortCAD utility objects and functions87
 - 5.1. ShortCADInputForm Object.....87
 - ShortCADInputForm Object Properties88
 - Constructor of ShortCADInputForm Object88
 - ShortCADInputForm.addInputField Method88
 - ShortCADInputForm.doModal Method.....90
 - 5.2. ShortCADYield function90
 - 5.3. ShortCADSleep function90

1. Introduction

This document is aimed at developers of CAD algorithms and programs that can run on ShortCAD and utilize its Drawing Object Model to create and modify ShortCAD drawings.

In this document you will find [ShortCAD JavaScript](#) API reference information, description of the ShortCAD Drawing Object Model (DrOM), and informative reusable code samples.

In this document [ShortCAD JavaScript](#) programs are referred as [ShortJS](#)- scripts, -commands, -libraries and so on.

[ShortCAD JavaScript](#) programming is based on using both, well known JavaScript objects and functions and the ones, provided to work with opened ShortCAD Drawing. This document does not consider standard JavaScript objects and functions like [Math](#), [Array](#), [String](#) and others.

The ShortCAD drawing object model, its objects and functions, provided to work with the drawings, are only the subjects, discussed here.

2. About ShortCAD JavaScript

ShortJS scripts are interactive programs which allow creating custom tools to create, access and modify contents of the ShortCAD drawings according to custom CAD algorithms.

ShortCAD Lite is provided for both, Mobile and PC Windows platforms. In fact, the PC environment is much more comfortable to code, try and test programs. Therefore, the process of developing of **ShortJS** scripts becomes much more exciting and effective when it's performed in PC environment, where specialized JavaScript editors, code analyzers, help systems, and other SW development tools are available.

ShortCAD JavaScript provides ECMAScript according to ECMA-262 and runs on ShortCAD Lite on both Windows CE/Windows Mobile and Windows XP/Vista/Windows 7 platforms.

ShortCAD JavaScript compiles and executes JavaScript source code, provides memory for created objects, and collects garbage objects, which are not used anymore.

Performance of ShortCAD JavaScript is one of the most key features of ShortCAD.

Since the **ShortJS** scripts are not embedded into the ShortCAD drawings, you can run the same script working with many different ShortCAD drawings. As well you can run many different scripts working with the same drawing. There is no limitation applied to order in which one **ShortJS** script can follow another when running.

All data of the opened ShortCAD Drawing are accessible via properties and methods of the object referred as **drawing**, which is instantiated at the start up time of the being launched **ShortJS** script. Using this object, **ShortJS** scripts get and set the drawing settings, access the existing and create new drawing objects like created lines, arcs, polylines etc. In their turn, the accessed/created drawing objects expose their specific properties and methods to allow analyzing, defining and modifying the objects. For example, it's possible to change endpoints of a line, radiuses of a circle, layer/color/linetype of any drawing object and so on.

For user interactivity, ShortCAD JavaScript provides special functions and object types to input and output data. The UI functions of ShortCAD JavaScript allow performing such actions as:

- Alerting with text like one made in the considered [HelloWorld.ShortJS](#),
- Interactive and careful entering of the points on drawing, distances and angles,
- Single and multiple selection of the existing drawing objects,
- Entering datasets with using programmable data input forms.

The version of ShortCAD JavaScript, which is considered in this document, is the first step. Yet, some of the definitions of ShortCAD drawings are not exposed to the level of running **ShortJS** scripts. Although, the available functionality is enough powerful to model, generate and interact with 2D Vector graphic contents.

3. Getting Started

Install ShortCAD Lite on your mobile device and optionally on PC. ShortCAD Lite installation packages are available at <http://www.shortcad.com>.

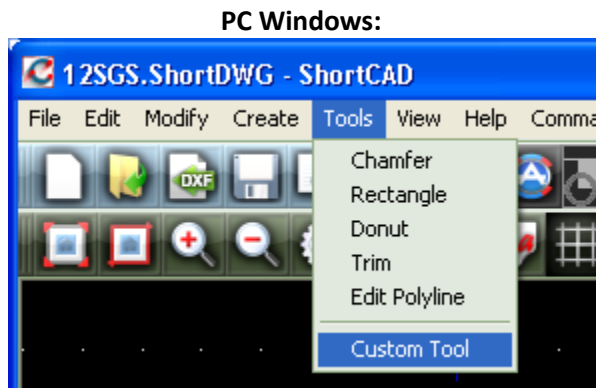
Using any text editor, create file *HelloWorld.ShortJS* with the following line:

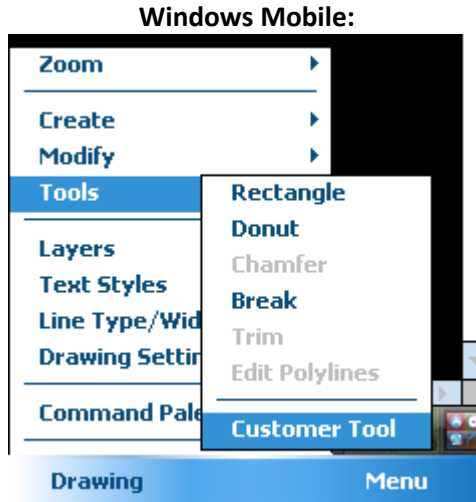
```
alert("Hello world!");
```

In case of Windows CE/Mobile locate *HelloWorld.ShortJS* in the folder *My Documents\ShortCAD Scripts*. You can choose other location. The only restriction is that *ShortJS* scripts should not be located in subfolders of subfolders. For example, standard open file dialog of Windows CE/Mobile will not "see" files in folder *My Documents\ShortCAD Scripts\Nuts Scripts*. When you will need a specially dedicated suborder, you should think about using folder *My Documents\Nuts Scripts*.

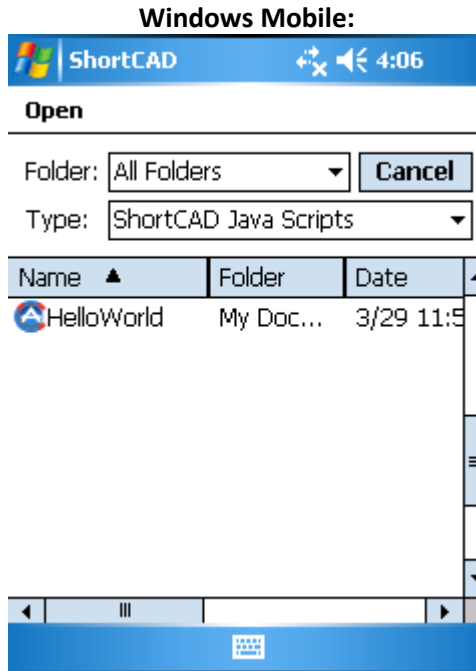
In case of using Windows for PC, use any location with no restrictions (e.g. a network path is useable).

Start ShortCAD Lite and tap menu Tools->Custom Tool.





In the *Open File* dialog, browse to the file *HelloWorld.ShortJS* and open it



As a result, you will see the message box saying **Hello World!**



If you have the message displayed, then you are ready to implement your algorithms and extend ShortCAD Lite with your own commands.

If you created and executed HelloWorld.ShortJS using PC, the only thing to do is to copy the file on your mobile device with installed ShortCAD Lite and make sure the script is running there as well.

4. ShortCAD Drawing Object model

4.1. WPoint Object

WPoint object represents an ordered pair of numeric x- and y-coordinates that defines a point in the World Coordinate System. X-axis is directed from left to right; Y-axis is directed from bottom to top.

WPoint Object Properties

Property	Description
<i>x</i>	<p>Number. Represents x-coordinate of the point. Direction of the X-axis is from left to right;</p> <p><i>Example:</i> <code>center.x = 10.5;</code></p>

<code>y</code>	<p>Number. Represents x-coordinate of the point. Direction of the X-axis is from left to right;</p> <p><i>Example:</i> <code>center.y = -15.2;</code></p>
----------------	---

Constructor of WPoint Object

`new WPoint()`

Creates new [WPoint](#) object with `WPoint.x = 0` and `WPoint.y = 0`.

`new WPoint(x, y)`

Creates new [WPoint](#) object with `WPoint.x = x` and `WPoint.y = y`.

WPoint.equals Method

`equals(otherPoint, tolerance)`

Returns `true` if `x` and `y` of the current point are equal to `x` and `y` of supplied point objects. Otherwise, returns `false`.

otherPoint - [WPoint](#) object to compare current point object with.

tolerance - Numeric value specifying the acceptable difference between the coordinates of the current point and *otherPoint* consider them as equal points. For example, if the *tolerance* is equal 0.001, then points [1, 2] and [1.001, 2.001] will be considered as equal.

If *tolerance* is not supplied, zero (0.0) value is used.

WPoint.distance Method

`distance (otherPoint)`

Returns distance between the current point object and supplied point object.

otherPoint - [WPoint](#) object to calculate the distance.

WPoint.angle Method

`angle(otherPoint)`

Returns angle in radians of vector, defined by the current point object (start of vector) and the supplied point object (end of vector).

otherPoint - [WPoint](#) object to calculate the angle of vector.

WPoint.rotate Method

`rotate(rotationCenter, rotationAngle)`

Rotates the current point object with the specified rotation center and angle.

rotationCenter - [WPoint](#) object specifying the rotation center.

rotationAngle - Numeric value specifying the rotation angle in radians.

WPoint.clone Method

`clone()`

Returns new created copy of the current point object.

4.2. DwgLAYER Object

`DwgLAYER` object represents a drawing layer. Opened drawing, represented by the `drawing` variable, contains at least one layer. This layer is named "0".

To retrieve collection of all the drawing layers, available in the opened drawing, call `drawing.getLayers()`.

To retrieve a particular drawing layer with a known name, call `drawing.getLayer(knownLayerName)`.

To create a new layer, call `drawing.createLayer(newLayerName)`.

DwgLAYER Object Properties

Property	Description
<i>name</i>	String. Name of the drawing layer.
<i>color</i>	Number. Integer index of color, applied to the drawing layer. To set <code>DwgLAYER.color</code> , one of the following values must be used: <ul style="list-style-type: none"> – an integer value in range [1...255], – <code>DwgColor.Default</code>.
<i>lineWeight</i>	Number. Line weight, applied to the drawing layer. To set <code>DwgLAYER.lineWeight</code> , one of the following values must be used: <ul style="list-style-type: none"> – a numeric value equal to or greater than zero (≥ 0.0), – <code>DwgLineWeight.Default</code>.
<i>lineType</i>	<code>DwgLINETYPE</code> . Line type, applied to the drawing layer. To set <code>DwgLAYER.lineType</code> , a <code>DwgLINETYPE</code> object, representing a drawing line type, existing in the opened drawing must be used.
<i>frozen</i>	Boolean. Indicates if the drawing layer is frozen (invisible). If <code>DwgLAYER.frozen</code> is <code>true</code> , the drawing layer is frozen.
<i>locked</i>	Boolean. Indicates if the drawing layer is locked (read-only). If <code>DwgLAYER.locked</code> is <code>true</code> , the drawing layer is locked.

4.3. DwgLINETYPE Object

DwgLINETYPE object represents a drawing line type.

The [currently opened drawing](#), represented by the [drawing](#) variable, contains at least one line type. This line type is named “CONTINUOUS”.

All available line types are defined in the file [ShortCAD.lts](#), located in the folder where ShortCAD is installed.

To retrieve collection of all the drawing line types, available in the opened drawing, call [drawing.getLineTypes\(\)](#).

To retrieve a particular drawing line type by its name, call [drawing.getLineType\(lineTypeName \)](#).

DwgLINETYPE Object Properties

Property	Description
<i>name</i>	String. Name of the drawing line type.

4.4. DwgTEXTSTYLE Object

DwgTEXTSTYLE object represents a drawing textstyle.

The [currently opened drawing](#), represented by the [drawing](#) variable, contains at least one text style. This text style is named "STANDARD".

To retrieve collection of all the drawing text styles, available in the opened drawing, call [drawing.getTextStyles\(\)](#).

To retrieve a particular drawing text style by its name, call [drawing.getTextStyle\(textStyleName \)](#).

DwgTEXTSTYLE Object Properties

Property	Description
<i>name</i>	String. Name of the drawing text style.
<i>fontName</i>	String. The name of a font applied to the drawing text style.
<i>height</i>	Number. Drawing text height, suggested by the text style. To set DwgTEXTSTYLE.height , a value equal to or greater than zero (≥ 0.0) must be used. Value 0.0 would mean that the text style does not suggest any text height.
<i>widthFactor</i>	Number. The drawing text relative X scale factor, suggested by the text style. To set DwgTEXTSTYLE.widthFactor , a value in range [0.01, 100.0] must be used.
<i>obliqueAngle</i>	Number. Oblique angle of the drawing text, supplied in radians.
<i>upsideDown</i>	Boolean. Indication if the text style suggests drawing text to be upside down (mirrored in Y). If DwgTEXTSTYLE.upsideDown is <code>true</code> , the drawing text is suggested to be mirrored in Y.
<i>backward</i>	Boolean. Indication if the text style suggests the drawing text to be backward (mirrored in X). If DwgTEXT.backward is <code>true</code> , the drawing text is suggested to be mirrored in X.

4.5. DwgObject Object

DwgObject is a general type of data which represents drawing objects (entities) like lines, arcs, circles and so on. It's used to work with the drawing objects when logic of algorithm (application) does not need to evaluate what exactly the kind of drawing object is. For example, there is no need to know what is being rotated, moved or erased. The following script shows, how generalization of the drawing object types could work:

```
for(
  var selected = drawing.selectObject("Select object?<quit>");
  selected instanceof DwgObject;
  selected = drawing.selectObject("Select object?<quit>")
)
{
  selected.color = DwgColor.Current;
}
```

In this example, `drawing.selectObject` prompts user to select an object and returns a *DwgObject* object when user taps one on the screen. If user presses Cancel, or starts closing application, or presses Enter-key as a default value, then `drawing.selectObject` returns a value of type different then *DwgObject*. This difference is used to check if the loop of changing color of selected by user elements shall be broken:

The statement “selected `instanceof` *DwgObject*” results with `false` if the returned variable `selected` is not an object of type *DwgObject*.

The following types of objects inherit all properties and methods of *DwgObject* object:

[DwgPOINT Object](#)

[DwgLINE Object](#)

[DwgSOLID Object](#)

[DwgTRACE Object](#)

[DwgARC Object](#)

[DwgCIRCLE Object](#)

[DwgELLIPSE Object](#)

[DwgPOLYLINE Object](#)

[DwgVERTEX Object](#)

[DwgINSERTION Object](#)

DwgObject Object Properties

Property	Description
----------	-------------

<i>color</i>	<p>Number. Integer index of color, applied to the current DwgObject. To set DwgObject.color, one of the following values must be used:</p> <ul style="list-style-type: none"> - an integer value in range [1...255], - DwgColor.Current, - DwgColor.Default, - DwgColor.ByLayer, - DwgColor.ByBlock.
<i>lineWeight</i>	<p>Number. Line weight, applied to the current DwgObject. To set DwgObject.lineWeight, one of the following values must be used:</p> <ul style="list-style-type: none"> - a numeric value equal to or greater than zero (≥ 0.0), - DwgLineWeight.Current, - DwgLineWeight.Default, - DwgLineWeight.ByLayer, - DwgLineWeight.ByBlock.
<i>lineType</i>	<p>DwgLINETYPE. Line type, applied to the current DwgObject. To set DwgObject.lineType, one of the following values must be used:</p> <ul style="list-style-type: none"> - DwgLINETYPE object representing a drawing line type, existing in the opened drawing, - DwgLineType.ByLayer, - DwgLineType.ByBlock, - DwgLineType.Current.
<i>layer</i>	<p>DwgLAYER. Layer, used to create new drawing objects. To set DwgObject.layer, one of the following must be used:</p> <ul style="list-style-type: none"> - DwgLAYER object representing a layer, existing in the opened drawing, - DwgLayer.Current.
<i>bounds</i>	<p>Object. Bounding rectangle expressing the extents of the drawing object represented by the current DwgObject.</p> <p><i>bounds</i> expresses the drawing object extents by means of the following two properties:</p> <ul style="list-style-type: none"> - <i>min</i> - WPoint object representing the bottom left corner of the bounding rectangle, - <i>max</i> - WPoint object representing the topright corner of the bounding rectangle.

DwgObject.draw Method

`draw(xorMode)`

Draws the current [DwgObject](#) on the screen applying XOR drawing mode if *xorMode* is `true`.

Returns the current [DwgObject](#).

Parameters:

XORMode - Boolean value specifying if the XOR mode should be applied. The mode is applied if *XORMode* is true.

DwgObject.move Method

move(displacement)

move(displacementX, displacementY)

Moves the current [DwgObject](#) with the specified displacement. The [DwgObject](#) for which *move* method is called, must either exist in the opened drawing or be created as an orphan drawing object. Moving of a drawing object does not automatically redraw it on the screen.

Returns the current [DwgObject](#).

Parameters:

- offset* - [WPoint](#) object specifies the displacement vector to move the current [DwgObject](#).
 - displacementX* -
 - displacementY* - Numeric values specifying *x* and *y* of the displacement vector to move the current [DwgObject](#).
-

DwgObject.rotate Method

rotate(rotationCenter, rotationAngle)

Rotates the current [DwgObject](#) with the specified rotation center and angle. Rotation of a drawing object does not automatically redraw it on the screen. Returns the current [DwgObject](#).

Parameters:

- rotationCenter* - [WPoint](#) object specifying the rotation center.
 - rotationAngle* - Numeric value specifying the rotation angle in radians.
-

DwgObject.scale Method

scale(scalingBase, scaleFactor)

Scales the current [DwgObject](#) with the specified scaling base point and scale factor. Scaling of a drawing object does not automatically redraw it on the screen.

Returns the current [DwgObject](#).

Parameters:

scalingBase - [WPoint](#) object specifying the base for scaling.

scaleFactor - Numeric value specifies the scale factor.

DwgObject.erase Method

erase()

Erases the current [DwgObject](#) from the drawing. The [DwgObject](#), for which *erase* method is called, must exist in the opened drawing. Erasing a drawing object will not automatically redraw the opened drawing on the screen.

Returns the current [DwgObject](#).

DwgObject.forget Method

forget()

This method is obsolete and has no effect. It is provided for backward compatibility. All unused orphan [DwgObjects](#) are automatically deleted from memory by garbage collector of [ShortCAD JavaScript](#).

Returns the current [DwgObject](#).

DwgObject.createCopy Method

createCopy(blockName)

Creates copy of a drawing object represented by the current [DwgObject](#) and, when specified, adds the created copy to the collection of drawing objects of a drawing block.

Returns the created [DwgObject](#).

Parameters:

blockName - **Optional** *String* object specifying the name of a drawing block to which collection of the drawing objects the created copy of drawing object should be added

When supplied, *blockName* must be one of the following:

- The name of a drawing block in a context of the [currently opened drawing](#). If the block with the specified name does not exist, it will be created with an empty collection of the drawing objects.

- `DwgBlock.OrphanBlock` to create drawing object copy which does not belong to the opened [drawing](#) (orphan). As an example, such drawing objects are useful for on screen echoing when indicating an intention to have finally created drawing contents.
- `DwgBlock.Drawing` to add the created drawing object copy to the end of the opened drawing body.

If *blockName* is not supplied, `DwgBlock.Drawing` is used.

DwgObject.intersections Method

`intersections(otherDwgObject1, otherDwgObject2, ... otherDwgObjectN)`

Finds all intersections of the current [DwgObject](#) with the one or more supplied [DwgObject](#)-s and returns an [Array](#) of the [WPoint](#) objects for the found intersections. When there is no intersections found, the returned [Array](#) object contains zero elements.

Returns [Array](#) of the [WPoint](#) objects.

Parameters:

otherDwgObject1 - [DwgObject](#) (at least 1 required) to find intersections of the current [DwgObject](#) with.

otherDwgObject2 -

...

otherDwgObjectN - **Optional** [DwgObjects](#) to find intersections of the current [DwgObject](#) with.

DwgObject.sameAs Method

`sameAs(otherDwgObject)`

Returns `true` if the current [DwgObject](#) and the supplied [DwgObject](#) represent the same drawing object. Otherwise, the method returns `false`;

Parameters:

otherDwgObject - [DwgObject](#) to evaluate if the current [DwgObject](#) is the same.

4.6. DwgPOINT Object

DwgPOINT object represents a drawing point. The way, drawing point objects are drawn on the screen or printed copy, depends on [drawing.pointDisplayMode](#) and [drawing.pointDisplaySize](#).

DwgPOINT object is derived from the [DwgObject](#) object.

DwgPOINT Object Properties

Property	Description
<i>location</i>	<p>WPoint. In the drawing space, location of the drawing point. Setting of DwgPOINT.location.x and DwgPOINT.location.y separately will have no effect. To change DwgPOINT.location, set it with another WPoint object.</p> <p><i>Example:</i></p> <pre>while((object = drawing.selectObject("Select POINT object?<quit>")) instanceof DwgPOINT) { // remove rendered raster of the point object from screen object.draw(true); //Swab x- and y- of the point object location var changedLocation = object.location; var xCoord = changedLocation.x changedLocation.x = changedLocation.y; changedLocation.y = xCoord; object.location = changedLocation; // render the of the modified point object on screen object.draw(true); }</pre>

In addition, [DwgPOINT](#) object inherits all properties the [DwgObject](#) object type. See: [DwgObject Properties](#).

DwgPOINT Object Methods

[DwgPOINT](#) object inherits all methods of the [DwgObject](#) object. See:

[DwgObject.draw](#), [DwgObject.move](#), [DwgObject.rotate](#), [DwgObject.scale](#),
[DwgObject.erase](#), [DwgObject.forget](#), [DwgObject.createCopy](#), [DwgObject.intersections](#)
and [DwgObject.sameAs](#).

4.7. DwgLINE Object

DwgLINE object represents a drawing straight line.

DwgLINE object is derived from the [DwgObject](#) object.

DwgLINE Object Properties

Property	Description
<i>start</i>	<p>WPoint. In the drawing space, location of the start point of the drawing line.</p> <p>Setting of DwgLINE.start.x and DwgLINE.start.y separately will have no effect. To change DwgLINE.start, set it with another WPoint object.</p> <p><i>Example:</i></p> <pre>while((object = drawing.selectObject("Select LINE object?<quit>")) instanceof DwgLINE) { // remove rendered raster of the line object from screen object.draw(true); //Swab x- and y- of the line object start var changedStart = object.start; var xCoord = changedStart.x changedStart.x = changedStart.y; changedStart.y = xCoord; object.start = changedStart; // render the modified line object on screen object.draw(false); }</pre>
<i>end</i>	<p>WPoint. In the drawing space, location of the end point of the drawing line.</p> <p>Setting of DwgLINE.end.x and DwgLINE.end.y separately will have no effect. To change DwgLINE.end, set it with another WPoint object.</p> <p><i>Example:</i></p> <pre>while((object = drawing.selectObject("Select LINE object?<quit>")) instanceof DwgLINE) { // remove rendered raster of the line object from screen</pre>

```
object.draw(true);

//Swab x- and y- of the line object end
var changedEnd = object.end;
var xCoord = changedEnd.x;
changedEnd.x = changedEnd.y;
changedEnd.y = xCoord;
object.end = changedEnd;

// render the modified line object on screen
object.draw(false);
}
```

In addition, [DwgLINE](#) object inherits all properties the [DwgObject](#) object type. See: [DwgObject Properties](#).

DwgLINE Object Methods

[DwgLINE](#) object inherits all methods of the [DwgObject](#) object. See:

[DwgObject.draw](#), [DwgObject.move](#), [DwgObject.rotate](#), [DwgObject.scale](#),
[DwgObject.erase](#), [DwgObject.forget](#), [DwgObject.createCopy](#), [DwgObject.intersections](#)
and [DwgObject.sameAs](#).

4.8. DwgSOLID Object

DwgSOLID object represents a drawing solid.

DwgSOLID object is derived from the [DwgObject](#) object.

DwgSOLID Object Properties

Property	Description
<i>firstCorner</i>	<p>WPoint. In the drawing space, location of the first corner of the drawing solid.</p> <p>Setting of DwgSOLID.firstCorner.x and DwgSOLID.firstCorner.y separately will have no effect. To change DwgSOLID.firstCorner, set it with another WPoint object.</p> <p><i>Example:</i></p> <pre>while((object = drawing.selectObject("Select SOLID object?<quit>")) instanceof DwgSOLID) { // remove rendered raster of the solid object from screen object.draw(true); //Swab x- and y- of the solid object first corner var changedPoint = object.firstCorner; var xCoord = changedPoint.x changedPoint.x = changedPoint.y changedPoint.y = xCoord; object.firstCorner = changedPoint; // render the modified solid object on screen object.draw(false); }</pre>
<i>secondCorner</i>	<p>WPoint. In the drawing space, location of the second corner of the drawing solid.</p> <p>Setting of DwgSOLID.secondCorner.x and DwgSOLID.secondCorner.y separately will have no effect. To change DwgSOLID.secondCorner, set it with another WPoint object.</p> <p><i>Example:</i></p> <pre>while((object = drawing.selectObject("Select SOLID object?<quit>")) instanceof DwgSOLID) { // remove rendered raster of the solid object from screen</pre>

	<pre> object.draw(true); //Swab x- and y- of the solid object second corner var changedPoint = object.secondCorner; var xCoord = changedPoint.x changedPoint.x = changedPoint.y changedPoint.y = xCoord; object.secondCorner = changedPoint; // render the modified solid object on screen object.draw(false); } </pre>
<i>thirdCorner</i>	<p>WPoint. In the drawing space, location of the third corner of the drawing solid.</p> <p>Setting of DwgSOLID.thirdCorner.x and DwgSOLID.thirdCorner.y separately will have no effect. To change DwgSOLID.thirdCorner, set it with another WPoint object.</p> <p><i>Example:</i></p> <pre> while((object = drawing.selectObject("Select SOLID object?<quit>")) instanceof DwgSOLID) { // remove rendered raster of the solid object from screen object.draw(true); //Swab x- and y- of the solid object third corner var changedPoint = object.thirdCorner; var xCoord = changedPoint.x changedPoint.x = changedPoint.y changedPoint.y = xCoord; object.thirdCorner = changedPoint; // render the modified solid object on screen object.draw(false); } </pre>
<i>fourthCorner</i>	<p>WPoint. In the drawing space, location of the fourth corner of the drawing solid.</p> <p>Setting of DwgSOLID.fourthCorner.x and DwgSOLID.fourthCorner.y separately will have no effect. To change DwgSOLID.fourthCorner, set it with another WPoint object.</p> <p><i>Example:</i></p> <pre> while((object = drawing.selectObject("Select SOLID object?<quit>")) instanceof DwgSOLID) { // remove rendered raster of the solid object from screen </pre>

	<pre>object.draw(true); //Swab x- and y- of the solid object fourth corner var changedPoint = object.fourthCorner; var xCoord = changedPoint.x changedPoint.x = changedPoint.y changedPoint.y = xCoord; object.fourthCorner = changedPoint; // render the modified solid object on screen object.draw(false); }</pre>
--	---

In addition, [DwgSOLID](#) object inherits all properties the [DwgObject](#) object type. See: [DwgObject Properties](#).

DwgSOLID Object Methods

[DwgSOLID](#) object inherits all methods of the [DwgObject](#) object. See:

[DwgObject.draw](#), [DwgObject.move](#), [DwgObject.rotate](#), [DwgObject.scale](#),
[DwgObject.erase](#), [DwgObject.forget](#), [DwgObject.createCopy](#), [DwgObject.intersections](#)
and [DwgObject.sameAs](#).

4.9. DwgTRACE Object

DwgTRACE object represents a drawing trace.

DwgTRACE object is derived from the [DwgObject](#) object.

DwgTRACE Object Properties

Property	Description
<i>firstCorner</i>	<p>WPoint. In the drawing space, location of the first corner of the drawing trace.</p> <p>Setting of DwgTRACE.firstCorner.x and DwgTRACE.firstCorner.y separately will have no effect. To change DwgTRACE.firstCorner, set it with another WPoint object.</p> <p><i>Example:</i></p> <pre>while((object = drawing.selectObject("Select TRACE object?<quit>")) instanceof DwgTRACE) { // remove rendered raster of the trace object from screen object.draw(true); //Swab x- and y- of the trace object first corner var changedPoint = object.firstCorner; var xCoord = changedPoint.x changedPoint.x = changedPoint.y changedPoint.y = xCoord; object.firstCorner = changedPoint; // render the modified trace object on screen object.draw(false); }</pre>
<i>secondCorner</i>	<p>WPoint. In the drawing space, location of the second corner of the drawing trace.</p> <p>Setting of DwgTRACE.secondCorner.x and DwgTRACE.secondCorner.y separately will have no effect. To change DwgTRACE.secondCorner, set it with another WPoint object.</p> <p><i>Example:</i></p> <pre>while((object = drawing.selectObject("Select TRACE object?<quit>")) instanceof DwgTRACE) { // remove rendered raster of the trace object from screen</pre>

	<pre> object.draw(true); //Swab x- and y- of the trace object second corner var changedPoint = object.secondCorner; var xCoord = changedPoint.x changedPoint.x = changedPoint.y changedPoint.y = xCoord; object.secondCorner = changedPoint; // render the modified trace object on screen object.draw(false); } </pre>
<p><i>thirdCorner</i></p>	<p>WPoint. In the drawing space, location of the third corner of the drawing trace.</p> <p>Setting of DwgTRACE.thirdCorner.x and DwgTRACE.thirdCorner.y separately will have no effect. To change DwgTRACE.thirdCorner, set it with another WPoint object.</p> <p><i>Example:</i></p> <pre> while((object = drawing.selectObject("Select TRACE object?<quit>")) instanceof DwgTRACE) { // remove rendered raster of the trace object from screen object.draw(true); //Swab x- and y- of the trace object third corner var changedPoint = object.thirdCorner; var xCoord = changedPoint.x changedPoint.x = changedPoint.y changedPoint.y = xCoord; object.thirdCorner = changedPoint; // render the modified trace object on screen object.draw(false); } </pre>
<p><i>fourthCorner</i></p>	<p>WPoint. In the drawing space, location of the fourth corner of the drawing trace.</p> <p>Setting of DwgTRACE.fourthCorner.x and DwgTRACE.fourthCorner.y separately will have no effect. To change DwgTRACE.fourthCorner, set it with another WPoint object.</p> <p><i>Example:</i></p> <pre> while((object = drawing.selectObject("Select TRACE object?<quit>")) instanceof DwgTRACE) { // remove rendered raster of the trace object from screen </pre>

```
object.draw(true);

//Swab x- and y- of the trace object fourth corner
var changedPoint = object.fourthCorner;
var xCoord = changedPoint.x
changedPoint.x = changedPoint.y
changedPoint.y = xCoord;
object.fourthCorner = changedPoint;

// render the modified trace object on screen
object.draw(false);
}
```

In addition, [DwgTRACE](#) object inherits all properties the [DwgObject](#) object type. See: [DwgObject Properties](#).

DwgTRACE Object Methods

[DwgTRACE](#) object inherits all methods of the [DwgObject](#) object. See:

[DwgObject.draw](#), [DwgObject.move](#), [DwgObject.rotate](#), [DwgObject.scale](#),
[DwgObject.erase](#), [DwgObject.forget](#), [DwgObject.createCopy](#), [DwgObject.intersections](#)
and [DwgObject.sameAs](#).

4.10. DwgARC Object

DwgARC object represents a drawing arc.

DwgARC object is derived from the [DwgObject](#) object.

DwgARC Object Properties

Property	Description
<i>center</i>	<p>WPoint. In the drawing space, location of the center of the drawing arc.</p> <p>Setting of DwgARC.center.x and DwgARC.center.y separately will have no effect. To change DwgARC.center, set it with another WPoint object.</p> <p><i>Example:</i></p> <pre>while((object = drawing.selectObject("Select ARC object?<quit>")) instanceof DwgARC) { // remove rendered raster of the arc object from screen object.draw(true); //Swab x- and y- of the arc object center var changedPoint = object.center; var xCoord = changedPoint.x changedPoint.x = changedPoint.y; changedPoint.y = xCoord; object.center = changedPoint; // render the modified arc object on screen object.draw(false); }</pre>
<i>radius</i>	<p>Number. Radius of the drawing arc. To set DwgARC.radius, a value greater than zero (> 0.0) must be used.</p>
<i>startAngle</i>	<p>Number. Start angle of the drawing arc, supplied in radians.</p>
<i>endAngle</i>	<p>Number. End angle of the drawing arc, supplied in radians.</p>

In addition, [DwgARC](#) object inherits all properties the [DwgObject](#) object type. See: [DwgObject Properties](#).

DwgARC Object Methods

[DwgARC](#) object inherits all methods of the [DwgObject](#) object. See:

[DwgObject.draw](#), [DwgObject.move](#), [DwgObject.rotate](#), [DwgObject.scale](#),
[DwgObject.erase](#), [DwgObject.forget](#), [DwgObject.createCopy](#), [DwgObject.intersections](#)
and [DwgObject.sameAs](#).

4.11. DwgCIRCLE Object

DwgCIRCLE object represents a drawing circle.

DwgCIRCLE object is derived from the [DwgObject](#) object.

DwgCIRCLE Object Properties

Property	Description
<i>center</i>	<p>WPoint. In the drawing space, location of the center of the drawing circle.</p> <p>Setting of DwgCIRCLE.center.x and DwgCIRCLE.center.y separately will have no effect. To change DwgCIRCLE.center, set it with another WPoint object.</p> <p><i>Example:</i></p> <pre>while((object = drawing.selectObject("Select CIRCLE object?<quit>")) instanceof DwgCIRCLE) { // remove rendered raster of the circle object from screen object.draw(true); //Swab x- and y- of the circle object center var changedPoint = object.center; var xCoord = changedPoint.x changedPoint.x = changedPoint.y; changedPoint.y = xCoord; object.center = changedPoint; // render the modified circle object on screen object.draw(false); }</pre>
<i>radius</i>	<p>Number. Radius of the drawing circle. To set DwgCIRCLE.radius, a value greater than zero (> 0.0) must be used.</p>

In addition, [DwgCIRCLE](#) object inherits all properties the [DwgObject](#) object type. See: [DwgObject Properties](#).

DwgCIRCLE Object Methods

[DwgCIRCLE](#) object inherits all methods of the [DwgObject](#) object. See:

[DwgObject.draw](#),

[DwgObject.move](#),

[DwgObject.rotate](#),

[DwgObject.scale](#),

[DwgObject.erase](#),

[DwgObject.forget](#),

[DwgObject.createCopy](#),

[DwgObject.intersections](#)

and [DwgObject.sameAs](#).

4.12. DwgELLIPSE Object

DwgELLIPSE object represents a drawing ellipse.

DwgELLIPSE object is derived from the [DwgObject](#) object.

DwgELLIPSE Object Properties

Property	Description
<i>center</i>	<p>WPoint. In the drawing space, location of the center of the drawing ellipse.</p> <p> DwgELLIPSE.center is tightly bound to DwgELLIPSE.majorRadiusPoint. When DwgELLIPSE.center is set with a new value point DwgELLIPSE.majorRadiusPoint is changed automatically to provide the same major radius and rotation of the major axis of the ellipse. Such way when moving an ellipse from one location to another can be done by means of changing only DwgELLIPSE.center and does not require calculation and changing of DwgELLIPSE.minor2majorRatio.</p> <p>Setting of DwgELLIPSE.center.x and DwgELLIPSE.center.y separately will have no effect. To change DwgELLIPSE.center, set it with another WPoint object.</p> <p><i>Example:</i></p> <pre>while((object = drawing.selectObject("Select ELLIPSE object?<quit>")) instanceof DwgELLIPSE) { // remove rendered raster of the ellipse object from screen object.draw(true); //Swab x- and y- of the ellipse object center var changedPoint = object.center; var xCoord = changedPoint.x changedPoint.x = changedPoint.y; changedPoint.y = xCoord; object.center = changedPoint; // render the modified ellipse object on screen object.draw(false); }</pre>
<i>majorRadiusPoint</i>	<p>WPoint. In the drawing space, location of the major radius end point of the drawing ellipse.</p>

	 <p>DwgELLIPSE.majorRadiusPoint is tightly bound to DwgELLIPSE.center. When DwgELLIPSE.center is set with a new value point DwgELLIPSE.majorRadiusPoint is changed automatically to provide the same major radius and rotation of the major axis of the ellipse. Such way when moving an ellipse from one location to another can be done by means of changing only DwgELLIPSE.center and does not require calculation and changing of DwgELLIPSE.minor2majorRatio.</p> <p>Setting of DwgELLIPSE.majorRadiusPoint.x and DwgELLIPSE.majorRadiusPoint.y separately will have no effect. To change DwgELLIPSE.majorRadiusPoint, set it with another WPoint object.</p> <p><i>Example:</i></p> <pre>while((object = drawing.selectObject("Select ELLIPSE object?<quit>")) instanceof DwgELLIPSE) { // remove rendered raster of the ellipse object from screen object.draw(true); //Swab x- and y- of the ellipse object center var changedPoint = object.majorRadiusPoint; var xCoord = changedPoint.x changedPoint.x = changedPoint.y; changedPoint.y = xCoord; object.majorRadiusPoint = changedPoint; // render the modified ellipse object on screen object.draw(false); }</pre>
<i>minor2majorRatio</i>	Number. Ratio of minor axis to major axis of the drawing ellipse. To set DwgELLIPSE.minor2majorRatio , a value greater than zero (> 0.0) must be
<i>startAngle</i>	Number. Start angle of the drawing ellipse, supplied in radians.
<i>endAngle</i>	Number. End angle of the drawing ellipse, supplied in radians.

In addition, [DwgELLIPSE](#) object inherits all properties the [DwgObject](#) object type. See: [DwgObject Properties](#).

DwgELLIPSE Object Methods

[DwgELLIPSE](#) object inherits all methods of the [DwgObject](#) object. See:

[DwgObject.draw](#), [DwgObject.move](#), [DwgObject.rotate](#), [DwgObject.scale](#),
[DwgObject.erase](#), [DwgObject.forget](#), [DwgObject.createCopy](#), [DwgObject.intersections](#)
and [DwgObject.sameAs](#).


4.13. DwgPOLYLINE Object

DwgPOLYLINE object represents a drawing polyline.

DwgPOLYLINE object is derived from the [DwgObject](#) object.

DwgPOLYLINE object is a compound drawing object and should consist of at least two [DwgVERTEX](#) objects.

DwgPOLYLINE Object Properties

Property	Description
<i>closed</i>	Boolean. Indicates if the drawing polyline is closed. If DwgPOLYLINE.closed is <code>true</code> , the drawing polyline is closed.
<i>smoothingType</i>	<p>Curves and smoothing type of the drawing polyline. To set DwgPOLYLINE.smoothingType, one of the following values must be used:</p> <ul style="list-style-type: none"> – <code>DwgPolyline.NoSmoothing</code>, – <code>DwgPolyline.QuadraticBSpline</code>, – <code>DwgPolyline.CubicBSpline</code>. <p> ShortCAD Lite ver. 2.3 and earlier do not support setting of the DwgPOLYLINE.smoothingType property. The only option is to read it. For example, expression like <code>myPolyline.smoothingType = mySmoothingType</code> will generate appropriate exception; the same time the expression <code>thatSmoothingType = myPolyline.smoothingType</code> will work fine.</p>

In addition, [DwgPOLYLINE](#) object inherits all properties the [DwgObject](#) object type. See: [DwgObject Properties](#).

DwgPOLYLINE Object Methods

In addition to its own methods, [DwgPOLYLINE](#) object inherits all methods of the [DwgObject](#) object. See:

[DwgObject.draw](#), [DwgObject.move](#), [DwgObject.rotate](#), [DwgObject.scale](#),
[DwgObject.erase](#), [DwgObject.forget](#), [DwgObject.createCopy](#), [DwgObject.intersections](#)
and [DwgObject.sameAs](#).

DwgPolyline.getVertexes Method

getVertexes()

Returns [Array](#) object consisting of the [DwgVERTEX](#) objects representing vertexes, which are composing the drawing polyline.

Adding or removing [DwgVERTEX](#) objects to or from the [Array](#) object returned by [DwgPOLYLINE.getVertex](#) method has no effect on the drawing polyline represented by the [DwgPOLYLINE](#) object.

It's recommended to avoid iterative calls to [DwgPOLYLINE.getVertex](#) (for example in the for-, while- and do/while-loops) since every call to this method creates and builds new [Array](#) of the [DwgVERTEX](#) objects, what is bad for overall performance. To iterate through a polyline vertex collection, assign the collection to a variable and use that variable for the iteration.

Example:

```
while((object =
    drawing.selectObject("Select POLYLINE object?<quit>"))
    instanceof DwgPOLYLINE)
{
    vertexCollection = object.getVertexes()

    // Mark each polyline vertex with a red circle,
    // created in the opened drawing
    for(var i = 0; i < vertexCollection.length; i++)
    {
        drawing.createCircle(
            vertexCollection[i].location,
            5,
            DwgColor.Red
        );
    }
}
```

DwgPolyline.appendVertex Method

appendsVertexes(
 vertexLocation,
 startWidth,
 endWidth,
 vertexBulge,
 vertexType
)

Appends vertex to end of the drawing polyline, represented by the current [DwgPOLYLINE](#) object.

Returns the [DwgVERTEX](#) object, representing the appended vertex.

Parameters:

vertexLocation - [WPoint](#) object supplying location of the vertex to append.

startWidth - **Optional Numeric** value specifies start width of the vertex to append. Must be equal to or greater than zero (≥ 0.0).

If *startWidth* is not supplied, 0.0 is used.

startWidth must be supplied if *endWidth* or further parameters are supplied.

endWidth - **Optional Numeric** value specifies start width of the vertex to append. Must be equal to or greater than zero (≥ 0.0).

If *endWidth* is not supplied, 0.0 is used.

endWidth must be supplied if *vertexBulge* or further parameters are supplied.

vertexBulge - **Optional Numeric** value specifies bulge of the vertex to append. Must be greater than or equal to zero.

If *vertexBulge* is not supplied, 0.0 is used.

vertexBulge must be supplied if *vertexType* is supplied.

vertexType - **Optional Numeric** integer value specifies type of the vertex to append.

vertexType be one of the following values:

- Zero value indicating ordinary vertex which is not a result of curve or spline fitting.
- [DwgPolyline.ControlPointVertex](#) indicating vertex which is a spline frame control point,
- [DwgPolyline.CurveFitPointVertex](#) indicating extra vertex created by curve-fitting,
- [DwgPolyline.SplineFitPointVertex](#) indicating vertex created by spline-fitting.

If *vertexType* is not supplied, 0 is used.

Example:

```
// Define a spiral and draw it with polyline packed with the vertexes
var spiralCenter = new WPoint(100, 150);
var spiralRotation = Math.PI / 6; // 30 degrees
var spiralStartRadius = 20;
var numberOfSpiralRotations = 4;
var spiralGrowthPerRotation = 15;
```

```
var spiralStepsPerRotation = 30;

var deltaRadius = spiralGrowthPerRotation / spiralStepsPerRotation;
var deltaAngle = (Math.PI * 2) / spiralStepsPerRotation;
var iterations = spiralStepsPerRotation * numberOfSpiralRotations + 1;

var radius = spiralStartRadius;
var angle = spiralRotation;

// Create POLYLINE
var polyline = drawing.createPolyline();

// Iteratively append the spiral vertexes to the polyline
for(var stepNumber = 0;
    stepNumber < iterations;
    stepNumber++)
{
    var vertexLocation = new WPoint(spiralCenter.x + radius * Math.cos(angle),
                                    spiralCenter.x + radius * Math.sin(angle));
    polyline.appendVertex(vertexLocation);
    radius += deltaRadius;
    angle += deltaAngle;
}
```

4.14. DwgVERTEX Object

DwgVERTEX object represents a vertex of a [drawing polyline](#).

DwgVERTEX object is derived from the [DwgObject](#) object.

DwgVERTEX Object Properties

Property	Description
<i>location</i>	<p>WPoint. In the drawing space, location of the vertex.</p> <p>Setting of DwgVERTEX.location.x and DwgVERTEX.location.y separately will have no effect. To change DwgVERTEX.location, set it with another WPoint object.</p> <p><i>Example:</i></p> <pre>while((object = drawing.selectObject("Select POLYLINE object?<quit>")) instanceof DwgPOLYLINE) { // remove rendered raster of the polyline object from screen object.draw(true); vertexCollection = object.getVertexes() //Swab x- and y- of each vertex of the polyline for(var i = 0; i < vertexCollection.length; i++) { var changedPoint = vertexCollection[i].location; var xCoord = changedPoint.x changedPoint.x = changedPoint.y; changedPoint.y = xCoord; vertexCollection[i].location = changedPoint; } // render the modified polyline object on screen object.draw(false); }</pre>
<i>startWidth</i>	<p>Number. Start width of the vertex.</p> <p>To set DwgVERTEX.startWidth, a value equal to or greater than zero (≥ 0.0) must be.</p>
<i>endWidth</i>	<p>Number. End width of the vertex.</p> <p>To set DwgVERTEX.endWidth, a value equal to or greater than zero (≥ 0.0) must be.</p>

<i>bulge</i>	<p>Number. Bulge of the vertex. The bulge is the tangent of $\frac{1}{4}$ of the included angle for an arc segment (e.g. bulge of 1.0 is a semicircle).</p> <p><i>bulge</i> is negative if the arc goes clockwise from the start point to the endpoint. A bulge of 0.0 means a straight segment.</p>
<i>type</i>	<p>Number. Integer value indicating type of the vertex.</p> <p>Must be one of the following values:</p> <ul style="list-style-type: none"> – Zero value indicating ordinary vertex which is not a result of curve or spline fitting. – DwgPolyline.ControlPointVertex indicating vertex which is a spline frame control point, – DwgPolyline.CurveFitPointVertex indicating extra vertex created by curve-fitting, – DwgPolyline.SplineFitPointVertex indicating vertex created by spline-fitting.

In addition, [DwgVERTEX](#) object inherits all properties the [DwgObject](#) object type. See: [DwgObject Properties](#).

DwgVERTEX Object Methods

[DwgVERTEX](#) object inherits all methods of the [DwgObject](#) object but the following:

- [draw](#),
- [forget](#),
- [intersections](#) and
- [erase](#).

See:

[DwgObject.move](#), [DwgObject.rotate](#), [DwgObject.scale](#), [DwgObject.createCopy](#)
and [DwgObject.sameAs](#).

4.15. DwgTEXT Object

DwgTEXT object represents a drawing text.

DwgTEXT object is derived from the [DwgObject](#) object.

DwgTEXT Object Properties

Property	Description
<i>text</i>	String. Text string applied to the drawing text and rendered on the screen when the text is visible.
<i>textStyle</i>	DwgTEXTSTYLE. Drawing text style applied to the drawing text. To set DwgTEXT.textStyle , one of the following must be used: <ul style="list-style-type: none"> – DwgTEXTSTYLE object representing a text style, existing in the opened drawing, – String object specifying the name of a text style, existing in the opened drawing, – DwgTextStyle.Current.
<i>height</i>	Number. Height of the drawing text. To set DwgTEXT.height , a value greater than zero (>0.0) must be used.
<i>widthFactor</i>	Number. Relative X scale factor of the drawing text. To set DwgTEXT.widthFactor , a value greater than zero (>0.0) must be used.
<i>obliqueAngle</i>	Number. Oblique angle of the drawing text, supplied in radians.
<i>rotation</i>	Number. Rotation angle of the drawing text, supplied in radians.
<i>upsideDown</i>	Boolean. Indication if the drawing text is upside down (mirrored in Y). If DwgTEXT.upsideDown is true , the drawing text is mirrored in Y.
<i>backward</i>	Boolean. Indication if the drawing text is backward (mirrored in X). If DwgTEXT.backward is true , the drawing text is mirrored in X.
<i>horzAlignment</i>	Number. Integer value, specifying the horizontal alignment of the drawing text. To set DwgTEXT.horzAlignment one of the following values must be used: <ul style="list-style-type: none"> – DwgTextStyle.LeftAligned, – DwgTextStyle.CenterAligned, – DwgTextStyle.RightAligned.

In addition, [DwgTEXT](#) object inherits all properties the [DwgObject](#) object type. See: [DwgObject Properties](#).

DwgTEXT Object Methods

[DwgTEXT](#) object inherits all methods of the [DwgObject](#) object. See:

[DwgObject.draw](#),

[DwgObject.move](#),

[DwgObject.rotate](#),

[DwgObject.scale](#),

[DwgObject.erase](#),

[DwgObject.forget](#),

[DwgObject.createCopy](#),

[DwgObject.intersections](#)

and [DwgObject.sameAs](#).

4.16. DwgINSERTION Object

DwgINSERTION object represents a reference to a drawing block (drawing symbols) inserted into the drawing body or another drawing block.

DwgINSERTION object is derived from the [DwgObject](#) object.

DwgINSERTION Object Properties

Property	Description
<i>location</i>	<p>WPoint. In the drawing space, insertion point of the drawing block reference.</p> <p>Setting of DwgINSERTION.location.x and DwgINSERTION.location.y separately will have no effect. To change DwgINSERTION.location, set it with another WPoint object.</p> <p><i>Example:</i></p> <pre>while((object = drawing.selectObject("Select INSERTION?<quit>")) instanceof DwgINSERTION) { // remove rendered raster of the insertion from screen object.draw(true); //Swab x- and y- of the insertion object center var changedPoint = object.location; var xCoord = changedPoint.x changedPoint.x = changedPoint.y; changedPoint.y = xCoord; object.location = changedPoint; // render the modified insertion on screen object.draw(false); }</pre>
<i>blockName</i>	<p>String. Name of the drawing block reference.</p> <p>To set DwgINSERTION.blockName, a String object specifying the name of a text style, existing in the opened drawing, must be used.</p>
<i>scaleX</i>	<p>Number. X scale factor of the inserted drawing block reference. To set DwgTEXTSTYLE.scaleX, a non-zero ($\neq 0.0$) must be used.</p>
<i>scaleY</i>	<p>Number. Y scale factor of the inserted drawing block reference. To set DwgTEXTSTYLE.scaleY, a non-zero ($\neq 0.0$) must be used.</p>
<i>rotation</i>	<p>Number. Rotation angle of the inserted drawing block reference, supplied in radians.</p>

In addition, [DwgTEXTSTYLE](#) object inherits all properties the [DwgObject](#) object type. See: [DwgObject Properties](#).

DwgINSERTION Object Methods

[DwgINSERTION](#) object inherits all methods of the [DwgObject](#) object. See:

[DwgObject.draw](#), [DwgObject.move](#), [DwgObject.rotate](#), [DwgObject.scale](#),
[DwgObject.erase](#), [DwgObject.forget](#), [DwgObject.createCopy](#), [DwgObject.intersections](#)
and [DwgObject.sameAs](#).

4.17. drawing Variable

`drawing` variable is of type `ShortCADDrawing` object.

`drawing` variable represents the currently opened ShortCAD Drawing. There is no way to create any object, other than `drawing`, to represent the currently opened ShortCAD Drawing. It does not mean that there is no way to assign `drawing` to another variable. For example, the following statement is absolutely correct:

```
var otherVarToReferOnOpenedDrawing = drawing;
```

drawing Variable Properties

Property	Description
<code>limitsMin</code>	<p>WPoint. Lower-left corner of drawing limits.</p> <p>Setting <code>drawing.limitsMin.x</code> and <code>drawing.limitsMin.y</code> separately will have no effect. To change <code>drawing.limitsMin</code>, set it with new WPoint object.</p> <p><i>Example:</i></p> <pre>function increaseDrawingSpace(extraWidth, extraHeight) { // Get current limitsMin var inflatedLimitsMin = drawing.limitsMin; // Subtract the extras from the minimum inflatedLimitsMin.x -= extraWidth / 2; inflatedLimitsMin.y -= extraHeight / 2; // Set drawing limits with the new minimum drawing.limitsMin = inflatedLimitsMin; }</pre>
<code>limitsMax</code>	<p>WPoint. Upper-right corner of drawing limits.</p> <p>Setting <code>drawing.limitsMax.x</code> and <code>drawing.limitsMax.y</code> separately will have no effect. To change <code>drawing.limitsMax</code>, set it with new WPoint object.</p> <p><i>Example:</i></p> <pre>. // Get current limitsMax var inflatedLimitsMax = drawing.limitsMax; // Add the extras to the maximum inflatedLimitsMax.x += extraWidth / 2; inflatedLimitsMax.y += extraHeight / 2;</pre>

	<pre> // Set drawing limits with the new maximum drawing.limitsMax = inflatedLimitsMax; } </pre>
<i>viewOrigin</i>	<p>WPoint. Lower-left corner of the currently viewed portion of drawing.</p> <p>Setting drawing.viewOrigin.x and drawing.viewOrigin.y separately will have no effect. To change drawing.viewOrigin, set it with new WPoint object.</p> <p><i>Example:</i></p> <pre> function increaseDrawingView(extraWidth, extraHeight) { // Get current viewOrigin var inflatedViewOrigin = drawing.viewOrigin; // Subtract the extras from the view origin inflatedViewOrigin.x -= extraWidth / 2; inflatedViewOrigin.y -= extraHeight / 2; // Set drawing view with the new origin drawing.viewOrigin = inflatedViewOrigin; } </pre>
<i>viewSize</i>	<p>WPoint. Size of the currently viewed portion of drawing.</p> <p>Setting drawing.viewSize.x and drawing.viewSize.y separately will have no effect. To change drawing.viewSize, set it with new WPoint object.</p> <p><i>Example:</i></p> <pre> // Get current viewSize var inflatedViewSize = drawing.viewSize; // Add the extras to the viewSize inflatedViewSize.x += extraWidth; inflatedViewSize.y += extraHeight; // Set drawing view with the new size drawing.viewSize = inflatedViewSize; } </pre>
<i>basePoint</i>	<p>WPoint. Insertion base point used when the drawing is inserted into other drawings.</p> <p>Setting drawing.basePoint.x and drawing.basePoint.y separately will have no effect. To change drawing.basePoint, set it with new WPoint object.</p> <p><i>Example:</i></p>

	<pre>// Set insertion base equal to the drawing limits minimum drawing.basePoint = drawing.limitsMin;</pre>
<i>gridMode</i>	<p>Boolean. The current visibility of a dot grid on viewed the drawing. The grid is shown if <i>drawing.gridMode</i> is <code>true</code></p>
<i>gridStep</i>	<p>Number. The current spacing of the dot grid, viewed on the drawing when <i>drawing.gridMode</i> is <code>true</code>. <i>drawing.gridStep</i> must be greater than or equal to zero (≥ 0.0).</p>
<i>color</i>	<p>Number. The current integer index of color, used to create new drawing objects. <i>drawing.color</i> must be one of the following values:</p> <ul style="list-style-type: none"> - an integer value in range [1...255], - <code>DwgColor.Default</code>, - <code>DwgColor.ByLayer</code>, - <code>DwgColor.ByBlock</code>.
<i>gridSnapMode</i>	<p>Boolean. The current restriction of the cursor movements to specified intervals (grid snap mode). The movements are restricted with the intervals specified by <i>drawing.gridSnapStep</i> if <i>drawing.gridSnapMode</i> is <code>true</code>.</p>
<i>gridSnapStep</i>	<p>Number. The current spacing of the grid snap mode. <i>drawing.gridSnapStep</i> must be a positive or zero value. Zero value is interpreted as grid snap mode, switched off. <i>drawing.gridSnapStep</i> must be greater than or equal to zero (≥ 0.0).</p>
<i>objectSnapMode</i>	<p>Boolean. The current restriction of the cursor movements to characteristic points on drawing objects (object snap mode). The movements are restricted to the object points if <i>drawing.objectSnapMode</i> is <code>true</code>.</p>
<i>orthoMode</i>	<p>Boolean. Constraining of the cursor movements to the horizontal or vertical direction (orthogonal snap mode). The movements are constrained with the directions if <i>drawing.orthoMode</i> is <code>true</code>.</p>
<i>intersectionSnapMode</i>	<p>Boolean. Restriction of the cursor movements to points of intersection points of different drawing objects (intersection snap mode). The movements are restricted to the intersection points if <i>drawing.intersectionSnapMode</i> is <code>true</code>.</p>
<i>chamferDistanceA</i>	<p>Number. The current first chamfer distance, used to chamfer drawing objects. <i>drawing.chamferDistanceA</i> must be greater than or equal to zero (≥ 0.0).</p>
<i>chamferDistanceB</i>	<p>Number. The current second chamfer distance, used to chamfer drawing objects. <i>drawing.chamferDistanceB</i> must be greater than or equal to zero (≥ 0.0).</p>
<i>polylineWidth</i>	<p>Number. The current polyline width, used to create new polylines.</p>

	<code>drawing.polylineWidth</code> must be greater than or equal to zero (≥ 0.0).
<code>traceWidth</code>	Number. The current trace width, used to create new width. <code>drawing.traceWidth</code> must be greater than or equal to zero (≥ 0.0).
<code>filletRadius</code>	Number. The current fillet radius, used to fillet drawing objects. <code>drawing.filletRadius</code> must be greater than or equal to zero (≥ 0.0).
<code>lineWeight</code>	Number. The current line weight, used to create new drawing objects. To set <code>drawing.lineWeight</code> , one of the following values must be used: <ul style="list-style-type: none"> - a numeric value equal to or greater than zero (≥ 0.0), - <code>DwgLineWeight.Default</code>, - <code>DwgLineWeight.ByLayer</code>, - <code>DwgLineWeight.ByBlock</code>.
<code>lineType</code>	<code>DwgLINETYPE</code> . The current drawing line type, used to create new drawing objects. To set <code>drawing.lineType</code> , one of the following values must be used: <ul style="list-style-type: none"> - <code>DwgLINETYPE</code> object, representing a drawing line type existing in the opened drawing, - String specifying the name of a drawing line type existing in the opened drawing, - <code>DwgLineType.ByLayer</code>, - <code>DwgLineType.ByBlock</code>.
<code>lineTypeScale</code>	Number. The current line type scale, currently used to render the drawing. <code>drawing.lineTypeScale</code> must be greater than zero (> 0.0).
<code>insideDonutDiameter</code>	Number. The current inside donut radius, used to create new donuts. <code>drawing.insideDonutDiameter</code> must be greater than or equal to zero (≥ 0.0).
<code>outsideDonutDiameter</code>	Number. The current outside donut radius, used to create new donuts. <code>drawing.outsideDonutDiameter</code> must be greater than or equal to zero (≥ 0.0).
<code>attributeDisplayMode</code>	<i>reserved for future use</i>
<code>pointDisplayMode</code>	Number. The current point display mode, used to render drawing point objects. <code>drawing.pointDisplayMode</code> must be one of the following:

	<div style="display: flex; flex-wrap: wrap; justify-content: space-around; text-align: center;"> <div style="margin: 5px;">.</div> <div style="margin: 5px;">+</div> <div style="margin: 5px;">×</div> <div style="margin: 5px;"> </div> </div> <div style="display: flex; justify-content: space-around; margin-top: 5px;"> 0 1 2 3 4 </div> <div style="display: flex; flex-wrap: wrap; justify-content: space-around; margin-top: 10px;"> <div style="margin: 5px;">⊙</div> <div style="margin: 5px;">○</div> <div style="margin: 5px;">⊕</div> <div style="margin: 5px;">⊗</div> <div style="margin: 5px;">⦶</div> </div> <div style="display: flex; justify-content: space-around; margin-top: 5px;"> 32 33 34 35 36 </div> <div style="display: flex; flex-wrap: wrap; justify-content: space-around; margin-top: 10px;"> <div style="margin: 5px;">◻</div> <div style="margin: 5px;">□</div> <div style="margin: 5px;">⊞</div> <div style="margin: 5px;">⊗</div> <div style="margin: 5px;">◻</div> </div> <div style="display: flex; justify-content: space-around; margin-top: 5px;"> 64 65 66 67 68 </div> <div style="display: flex; flex-wrap: wrap; justify-content: space-around; margin-top: 10px;"> <div style="margin: 5px;">◻</div> <div style="margin: 5px;">□</div> <div style="margin: 5px;">⊞</div> <div style="margin: 5px;">⊗</div> <div style="margin: 5px;">◻</div> </div> <div style="display: flex; justify-content: space-around; margin-top: 5px;"> 96 97 98 99 100 </div>
<i>pointDisplaySize</i>	Number. The current point display size, used to render drawing point objects.
<i>layer</i>	<p>DwgLAYER. The current layer, used to create new drawing objects. To set drawing.layer, one of the following values must be used:</p> <ul style="list-style-type: none"> – DwgLAYER object, representing a drawing layer existing in the opened drawing, – String specifying the name of a drawing layer existing in the opened drawing.
<i>textStyle</i>	DwgTEXTSTYLE . The current text style, used to create new text objects.

drawing.createPoint Method

```
createPoint(
    blockName,
    location,
    color,
    lineWeight,
    layer,
    lineType
)
```

Creates new drawing point represented by [DwgPOINT](#) object.

Returns [DwgPOINT](#) object, representing the created drawing point.

Parameters:

blockName - **Optional String** object specifying the name of a drawing block to which collection of the drawing objects the created drawing point should be added.

When supplied, *blockName* must be one of the following:

- The name of a drawing block in a context of the [currently opened drawing](#). If the block with the specified name does not exist, it will be created with an empty collection of the drawing objects,
- [DwgBlock.OrphanBlock](#) to create the drawing point which does not belong to the opened drawing (orphan). As an example, such drawing points are useful for on screen echoing when indicating an intention to have finally created drawing contents,
- [DwgBlock.Drawing](#) to add the created drawing point to the end of the opened drawing body.

If *blockName* is not supplied, [DwgBlock.Drawing](#) is used.

location

- [WPoint](#) object specifying location of the drawing point.

color

- **Optional** integer [Number](#) specifying the index of color to apply to the drawing point.

When supplied, *color* must be one of the following values:

- an integer value in range [1...255],
- [DwgColor.Current](#),
- [DwgColor.Default](#),
- [DwgColor.ByLayer](#),
- [DwgColor.ByBlock](#).

If *color* is not supplied, [DwgColor.Current](#) is used.

color must be supplied if *lineWeight* is supplied.

lineWeight

- **Optional** [Number](#) specifying the line weight to apply to the drawing point.

When supplied, *lineWeight* must be one of the following values:

- a numeric value equal to or greater than zero (≥ 0.0),
- [DwgLineWeight.Current](#),
- [DwgLineWeight.Default](#),
- [DwgLineWeight.ByLayer](#),
- [DwgLineWeight.ByBlock](#).

If *lineWeight* is not supplied, [DwgLineWeight.Current](#) is used.

lineWeight must be supplied if *layer* is supplied.

layer

- **Optional** [DwgLayer](#) object specifying the drawing layer to apply to the drawing point.

When supplied, *layer* must be one of the following values:

- [DwgLAYER](#) object, representing a drawing layer existing in the opened drawing,
- [String](#) specifying the name of a drawing layer existing in the opened drawing.
- [DwgLayer.Current](#).

If *layer* is not supplied, [DwgLayer.Current](#) is used.

layer must be supplied if *lineType* is supplied.

- lineType*
- **Optional** [DwgLINETYPE](#) object specifying the drawing line type to apply to the drawing point.

When supplied, *lineType* must be one of the following values:

- [DwgLINETYPE](#) object, representing a drawing line type existing in the opened drawing,
- [String](#) specifying the name of a drawing line type existing in the opened drawing,
- [DwgLineType.Current](#),
- [DwgLineType.ByLayer](#),
- [DwgLineType.ByBlock](#).

If *lineType* is not supplied, [DwgLineType.Current](#) is used.

drawing.createLine Method

```
createLine(
    blockName,
    startPoint,
    endPoint,
    color,
    lineWeight,
    layer,
    lineType
)
```

Creates new drawing line represented by [DwgLINE](#) object.

Returns [DwgLINE](#) object, representing the created drawing line.

Parameters:

- blockName*
- **Optional** [String](#) object specifying the name of a drawing block to which collection of the drawing objects the created drawing line should be added.

When supplied, *blockName* must be one of the following:

- The name of a drawing block in a context of the [currently opened drawing](#). If the block with the specified name does not exist, it will be created with an empty collection of the drawing objects,
- [DwgBlock.OrphanBlock](#) to create the drawing line which does not belong to the opened drawing (orphan). As an example, such lines are useful for on screen echoing when indicating an intention to have finally created drawing contents,
[DwgBlock.Drawing](#) to add the created drawing line to the end of the opened drawing body.

If *blockName* is not supplied, [DwgBlock.Drawing](#) is used.

- startPoint* - [WPoint](#) object specifying location of the start point of the drawing line.
- endPoint* - [WPoint](#) object specifying location of the end point of the drawing line.
- color* - **Optional** integer [Number](#) specifying the index of color to apply to the drawing line.

color must be one of the following values:

- an integer value in range [1...255],
- [DwgColor.Current](#),
- [DwgColor.Default](#),
- [DwgColor.ByLayer](#),
- [DwgColor.ByBlock](#).

If *color* is not supplied, [DwgColor.Current](#) is used.

color must be supplied if *lineWeight* is supplied.

- lineWeight* - **Optional** [Number](#) specifying the line weight to apply to the drawing line.

When supplied, *lineWeight* must be one of the following values:

- a numeric value equal to or greater than zero (≥ 0.0),
- [DwgLineWeight.Current](#),
- [DwgLineWeight.Default](#),
- [DwgLineWeight.ByLayer](#),
- [DwgLineWeight.ByBlock](#).

If *lineWeight* is not supplied, [DwgLineWeight.Current](#) is used.

lineWeight must be supplied if *layer* is supplied.

- layer* - **Optional** [DwgLayer](#) object specifying the drawing layer to apply to the drawing line.

When supplied, *layer* must be one of the following values:

- [DwgLAYER](#) object, representing a drawing layer existing in the opened drawing,
- *String* specifying the name of a drawing layer existing in the opened drawing.
- `DwgLayer.Current`.

If *layer* is not supplied, `DwgLayer.Current` is used.

layer must be supplied if *lineType* is supplied.

- lineType*
- **Optional** [DwgLINETYPE](#) object specifying the drawing line type to apply to the new drawing line.

lineType must be one of the following values:

- [DwgLINETYPE](#) object, representing a drawing line type existing in the opened drawing,
- *String* specifying the name of a drawing line type existing in the opened drawing,
- `DwgLineType.Current`,
- `DwgLineType.ByLayer`,
- `DwgLineType.ByBlock`.

If *lineType* is not supplied, `DwgLineType.Current` is used.

drawing.createPolyline Method

```
createPolyline(  
    blockName,  
    closed,  
    color,  
    lineWeight,  
    layer,  
    lineType  
)
```

Creates new drawing polyline represented by [DwgPOLYLINE](#) object.

Returns [DwgPOLYLINE](#) object, representing the created drawing polyline.

Parameters:

- blockName*
- **Optional** *String* object specifying the name of a drawing block to which collection of the drawing objects the created drawing polyline should be added.

When supplied, *blockName* must be one of the following:

- The name of a drawing block in a context of the [currently opened drawing](#). If the block with the specified name does not exist, it will be created with an empty collection of the drawing objects,
- [DwgBlock.OrphanBlock](#) to create the drawing polyline which does not belong to the opened drawing (orphan). As an example, such polylines are useful for on screen echoing when indicating an intention to have finally created drawing contents,
- [DwgBlock.Drawing](#) to add the created drawing polyline to the end of the opened drawing body.

If *blockName* is not supplied, [DwgBlock.Drawing](#) is used.

closed - **Optional Boolean** value specifying if the drawing polyline is closed. The polyline is closed if *closed* is `true`.

color - **Optional integer Number** specifying the index of color to apply to the drawing polyline.

color must be one of the following values:

- an integer value in range [1...255],
- [DwgColor.Current](#),
- [DwgColor.Default](#),
- [DwgColor.ByLayer](#),
- [DwgColor.ByBlock](#).

If *color* is not supplied, [DwgColor.Current](#) is used.

color must be supplied if *lineWeight* is supplied.

lineWeight - **Optional Number** specifying the line weight to apply to the drawing polyline.

When supplied, *lineWeight* must be one of the following values:

- a numeric value equal to or greater than zero (≥ 0.0),
- [DwgLineWeight.Current](#),
- [DwgLineWeight.Default](#),
- [DwgLineWeight.ByLayer](#),
- [DwgLineWeight.ByBlock](#).

If *lineWeight* is not supplied, [DwgLineWeight.Current](#) is used.

lineWeight must be supplied if *layer* is supplied.

layer - **Optional [DwgLayer](#)** object specifying the drawing layer to apply to the drawing polyline.

When supplied, *layer* must be one of the following values:

- [DwgLAYER](#) object, representing a drawing layer existing in the opened drawing,
- *String* specifying the name of a drawing layer existing in the opened drawing.
- `DwgLayer.Current`.

If *layer* is not supplied, `DwgLayer.Current` is used.

layer must be supplied if *lineType* is supplied.

lineType

- **Optional** [DwgLINETYPE](#) object specifying the drawing line type to apply to the new drawing polyline.

lineType must be one of the following values:

- [DwgLINETYPE](#) object, representing a drawing line type existing in the opened drawing,
- *String* specifying the name of a drawing line type existing in the opened drawing,
- `DwgLineType.Current`,
- `DwgLineType.ByLayer`,
- `DwgLineType.ByBlock`.

If *lineType* is not supplied, `DwgLineType.Current` is used.

drawing.createSolid Method

```
createSolid(  
    blockName,  
    firstCorner,  
    secondCorner,  
    thirdCorner,  
    fourthCorner,  
    color,  
    lineWeight,  
    layer,  
    lineType  
)
```

Creates new drawing solid-filled object represented by [DwgSOLID](#) object.

Returns [DwgSOLID](#) object, representing the created drawing solid-filled object.

Parameters:

blockName - **Optional String** object specifying the name of a drawing block to which collection of the drawing objects the created drawing solid-filled object should be added.

When supplied, *blockName* must be one of the following:

- The name of a drawing block in a context of the [currently opened drawing](#). If the block with the specified name does not exist, it will be created with an empty collection of the drawing objects,
- [DwgBlock.OrphanBlock](#) to create the drawing solid-filled object which does not belong to the opened drawing (orphan). As an example, such solid-filled objects are useful for on screen echoing when indicating an intention to have finally created drawing contents,
- [DwgBlock.Drawing](#) to add the created drawing solid-filled object to the end of the opened drawing body.

If *blockName* is not supplied, [DwgBlock.Drawing](#) is used.

firstCorner - [WPoint](#) object specifying location of the first corner of the drawing solid-filled object.

secondCorner - [WPoint](#) object specifying location of the second corner of the drawing solid-filled object.

thirdCorner - [WPoint](#) object specifying location of the third corner of the drawing solid-filled object.

fourthCorner - [WPoint](#) object specifying location of the fourth corner of the drawing solid-filled object.

color - **Optional integer Number** specifying the index of color to apply to the drawing solid-filled object.

color must be one of the following values:

- an integer value in range [1...255],
- [DwgColor.Current](#),
- [DwgColor.Default](#),
- [DwgColor.ByLayer](#),
- [DwgColor.ByBlock](#).

If *color* is not supplied, [DwgColor.Current](#) is used.

color must be supplied if *lineWeight* is supplied.

lineWeight - **Optional Number** specifying the line weight to apply to the drawing solid-filled object.

When supplied, *lineWeight* must be one of the following values:

- a numeric value equal to or greater than zero (≥ 0.0),
- `DwgLineWeight.Current`,
- `DwgLineWeight.Default`,
- `DwgLineWeight.ByLayer`,
- `DwgLineWeight.ByBlock`.

If *lineWeight* is not supplied, `DwgLineWeight.Current` is used.

lineWeight must be supplied if *layer* is supplied.

layer

- **Optional** `DwgLayer` object specifying the drawing layer to apply to the drawing solid-filled object.

When supplied, *layer* must be one of the following values:

- `DwgLAYER` object, representing a drawing layer existing in the opened drawing,
- `String` specifying the name of a drawing layer existing in the opened drawing.
- `DwgLayer.Current`.

If *layer* is not supplied, `DwgLayer.Current` is used.

layer must be supplied if *lineType* is supplied.

lineType

- **Optional** `DwgLINETYPE` object specifying the drawing line type to apply to the new drawing solid-filled object.

lineType must be one of the following values:

- `DwgLINETYPE` object, representing a drawing line type existing in the opened drawing,
- `String` specifying the name of a drawing line type existing in the opened drawing,
- `DwgLineType.Current`,
- `DwgLineType.ByLayer`,
- `DwgLineType.ByBlock`.

If *lineType* is not supplied, `DwgLineType.Current` is used.

drawing.createTrace Method

```
createTrace(  
    blockName,  
    firstCorner,  
    secondCorner,  
    thirdCorner,
```

```

    fourthCorner,
    color,
    lineWeight,
    layer,
    lineType
  )

```

Creates new drawing trace (solid line) represented by [DwgTRACE](#) object.

Returns [DwgTRACE](#) object, representing the created drawing trace.

Parameters:

blockName - **Optional String** object specifying the name of a drawing block to which collection of the drawing objects the created drawing trace should be added.

When supplied, *blockName* must be one of the following:

- The name of a drawing block in a context of the [currently opened drawing](#). If the block with the specified name does not exist, it will be created with an empty collection of the drawing objects,
- [DwgBlock.OrphanBlock](#) to create the drawing trace which does not belong to the opened drawing (orphan). As an example, such traces are useful for on screen echoing when indicating an intention to have finally created drawing contents,
- [DwgBlock.Drawing](#) to add the created drawing trace to the end of the opened drawing body.

If *blockName* is not supplied, [DwgBlock.Drawing](#) is used.

firstCorner - [WPoint](#) object specifying location of the first corner of the drawing trace.

secondCorner - [WPoint](#) object specifying location of the second corner of the drawing trace.

thirdCorner - [WPoint](#) object specifying location of the third corner of the drawing trace.

fourthCorner - [WPoint](#) object specifying location of the fourth corner of the drawing trace.

color - **Optional integer Number** specifying the index of color to apply to the drawing trace.

color must be one of the following values:

- an integer value in range [1...255],
- [DwgColor.Current](#),
- [DwgColor.Default](#),
- [DwgColor.ByLayer](#),
- [DwgColor.ByBlock](#).

If *color* is not supplied, `DwgColor.Current` is used.

color must be supplied if *lineWeight* is supplied.

lineWeight - **Optional Number** specifying the line weight to apply to the drawing trace.

When supplied, *lineWeight* must be one of the following values:

- a numeric value equal to or greater than zero (≥ 0.0),
- `DwgLineWeight.Current`,
- `DwgLineWeight.Default`,
- `DwgLineWeight.ByLayer`,
- `DwgLineWeight.ByBlock`.

If *lineWeight* is not supplied, `DwgLineWeight.Current` is used.

lineWeight must be supplied if *layer* is supplied.

layer - **Optional `DwgLayer`** object specifying the drawing layer to apply to the drawing trace.

When supplied, *layer* must be one of the following values:

- `DwgLAYER` object, representing a drawing layer existing in the opened drawing,
- `String` specifying the name of a drawing layer existing in the opened drawing.
- `DwgLayer.Current`.

If *layer* is not supplied, `DwgLayer.Current` is used.

layer must be supplied if *lineType* is supplied.

lineType - **Optional `DwgLINETYPE`** object specifying the drawing line type to apply to the new drawing trace.

lineType must be one of the following values:

- `DwgLINETYPE` object, representing a drawing line type existing in the opened drawing,
- `String` specifying the name of a drawing line type existing in the opened drawing,
- `DwgLineType.Current`,
- `DwgLineType.ByLayer`,
- `DwgLineType.ByBlock`.

If *lineType* is not supplied, `DwgLineType.Current` is used.

drawing.createArc Method

```
createArc(  
    blockName,  
    DwgArc.ByCenterRadiusAngles,  
    center,  
    radius,  
    startAngle,  
    endAngle,  
    color,  
    lineWeight,  
    layer,  
    lineType  
)
```

```
createArc(  
    blockName,  
    DwgArc.By3Points,  
    startPoint,  
    otherPoint,  
    endPoint,  
    color,  
    lineWeight,  
    layer,  
    lineType  
)
```

```
createArc(  
    blockName,  
    DwgArc.ByCenterStartEnd,  
    center,  
    startPoint,  
    endPoint,  
    color,  
    lineWeight,  
    layer,  
    lineType  
)
```

Creates new drawing arc represented by [DwgARC](#) object.

Returns [DwgARC](#) object, representing the created drawing arc.

Parameters:

[DwgArc.ByCenterRadiusAngles](#), [DwgArc.ByCenterStartEnd](#), and [DwgArc.By3Points](#), are the constants specifying in which way the drawing arc to create is defined.

[DwgArc.ByCenterRadiusAngles](#) is used when the arc to create is defined by its center, radius and the start- and counterclockwise end- angles.

`DwgArc.ByCenterStartEnd` is used when the arc to create is defined by its center, start point and counterclockwise end point.

`DwgArc.By3Points` is used when the arc to create is defined by its start point, counterclockwise end point and one point on the circumference of the arc. All the three points must be different to provide the arc definition consistency.

If no one of the constants is supplied the `DwgArc.ByCenterRadiusAngles` is used.

blockName - **Optional String** object specifying the name of a drawing block to which collection of the drawing objects the created drawing arc should be added.

When supplied, *blockName* must be one of the following:

- The name of a drawing block in a context of the [currently opened drawing](#). If the block with the specified name does not exist, it will be created with an empty collection of the drawing objects,
- `DwgBlock.OrphanBlock` to create the drawing arc which does not belong to the opened drawing (orphan). As an example, such arcs are useful for on screen echoing when indicating an intention to have finally created drawing contents,
`DwgBlock.Drawing` to add the created drawing arc to the end of the opened drawing body.

If *blockName* is not supplied, `DwgBlock.Drawing` is used.

center - [WPoint](#) object specifying location of the center of the drawing arc.

radius - **Number** specifying radius of the drawing arc.

radius must be greater than zero(>0.0).

startAngle - **Number** specifying start angle of the drawing arc, supplied in radians.

endAngle - **Number** specifying end angle of the drawing arc, supplied in radians.

startPoint - [WPoint](#) object specifying location of the start point of the drawing arc. When `DwgArc.ByCenterStartEnd` is used to create a drawing arc, the distance from *center* to *startPoint* defines the radius of the arc.

endPoint - [WPoint](#) object specifying location of the end point of the drawing arc.

otherPoint - [WPoint](#) object specifying location of a point on the circumference of the drawing arc.

color - **Optional integer Number** specifying the index of color to apply to the drawing arc.

color must be one of the following values:

- an integer value in range [1...255],
- `DwgColor.Current`,
- `DwgColor.Default`,
- `DwgColor.ByLayer`,
- `DwgColor.ByBlock`.

If *color* is not supplied, `DwgColor.Current` is used.

color must be supplied if *lineWeight* is supplied.

lineWeight - **Optional Number** specifying the line weight to apply to the drawing arc.

When supplied, *lineWeight* must be one of the following values:

- a numeric value equal to or greater than zero (≥ 0.0),
- `DwgLineWeight.Current`,
- `DwgLineWeight.Default`,
- `DwgLineWeight.ByLayer`,
- `DwgLineWeight.ByBlock`.

If *lineWeight* is not supplied, `DwgLineWeight.Current` is used.

lineWeight must be supplied if *layer* is supplied.

layer - **Optional `DwgLayer`** object specifying the drawing layer to apply to the drawing arc.

When supplied, *layer* must be one of the following values:

- `DwgLAYER` object, representing a drawing layer existing in the opened drawing,
- `String` specifying the name of a drawing layer existing in the opened drawing.
- `DwgLayer.Current`.

If *layer* is not supplied, `DwgLayer.Current` is used.

layer must be supplied if *lineType* is supplied.

lineType - **Optional `DwgLINETYPE`** object specifying the drawing line type to apply to the new drawing arc.

lineType must be one of the following values:

- `DwgLINETYPE` object, representing a drawing line type existing in the opened drawing,
- `String` specifying the name of a drawing line type existing in the opened drawing,

- `DwgLineType.Current`,
- `DwgLineType.ByLayer`,
- `DwgLineType.ByBlock`.

If *lineType* is not supplied, `DwgLineType.Current` is used.

drawing.createCircle Method

```
createCircle(  
    blockName,  
    DwgCircle.ByCenterRadius,  
    center,  
    radius,  
    color,  
    lineWeight,  
    layer,  
    lineType  
)
```

```
createCircle(  
    blockName,  
    DwgCircle.ByCenterRadius,  
    center,  
    radiusPoint,  
    color,  
    lineWeight,  
    layer,  
    lineType  
)
```

```
createCircle(  
    blockName,  
    DwgCircle.ByDiameter,  
    firstDiameterPoint,  
    otherDiameterPoint,  
    color,  
    lineWeight,  
    layer,  
    lineType  
)
```

```
createCircle(  
    blockName,  
    DwgCircle.By3Points,  
    firstPoint,  
    secondPoint,  
    thirdPoint,  
    color,  
    lineWeight,  
    layer,  
)
```

```

    lineType
  )

```

Creates new drawing circle represented by [DwgCIRCLE](#) object.

Returns [DwgCIRCLE](#) object, representing the created drawing circle.

Parameters:

[DwgCircle.ByCenterRadius](#), [DwgCircle.ByDiameter](#), and [DwgCircle.By3Points](#), are the constants specifying in which way the drawing circle to create is defined.

[DwgCircle.ByCenterRadius](#) is used when the circle to create is defined by its center and radius. The radius of the circle may be defined either by [Number](#) value or by a point on the circumference of the circle.

[DwgCircle.ByDiameter](#) is used when the circle to create is defined by the end points of a diameter of the circle.

[DwgCircle.By3Points](#) is used when the circle to create is defined by any three points on the circumference of the circle. All the three points must be different to provide the circle definition consistency.

If no one of the constants is supplied the [DwgCircle.ByCenterRadius](#) is used.

blockName - **Optional** [String](#) object specifying the name of a drawing block to which collection of the drawing objects the created drawing circle should be added.

When supplied, *blockName* must be one of the following:

- The name of a drawing block in a context of the [currently opened drawing](#). If the block with the specified name does not exist, it will be created with an empty collection of the drawing objects,
- [DwgBlock.OrphanBlock](#) to create the drawing circle which does not belong to the opened drawing (orphan). As an example, such circles are useful for on screen echoing when indicating an intention to have finally created drawing contents,
- [DwgBlock.Drawing](#) to add the created drawing circle to the end of the opened drawing body.

If *blockName* is not supplied, [DwgBlock.Drawing](#) is used.

center - [WPoint](#) object specifying location of the center of the drawing circle.

radius - [Number](#) specifying radius of the drawing circle.

radius must be greater than zero(>0.0).

- radiusPoint* - [WPoint](#) object specifying location of the end point of a radius of the drawing circle. The distance between *center* and *radiusPoint* is used as the radius of the circle.
- firstRadiusPoint* - [WPoint](#) object specifying location of the first end point of a diameter of the drawing circle.
- otherRadiusPoint* - [WPoint](#) object specifying location of the other end point of a diameter of the drawing circle.
- firstPoint* - [WPoint](#) object specifying location of the first point on the circumference of the drawing circle.
- secondPoint* - [WPoint](#) object specifying location of the second point on the circumference of the drawing circle.
- thirdPoint* - [WPoint](#) object specifying location of the first point on the circumference of the drawing circle.
- color* - **Optional** integer [Number](#) specifying the index of color to apply to the drawing circle.

color must be one of the following values:

- an integer value in range [1...255],
- [DwgColor.Current](#),
- [DwgColor.Default](#),
- [DwgColor.ByLayer](#),
- [DwgColor.ByBlock](#).

If *color* is not supplied, [DwgColor.Current](#) is used.

color must be supplied if *lineWeight* is supplied.

- lineWeight* - **Optional** [Number](#) specifying the line weight to apply to the drawing circle.

When supplied, *lineWeight* must be one of the following values:

- a numeric value equal to or greater than zero (≥ 0.0),
- [DwgLineWeight.Current](#),
- [DwgLineWeight.Default](#),
- [DwgLineWeight.ByLayer](#),
- [DwgLineWeight.ByBlock](#).

If *lineWeight* is not supplied, [DwgLineWeight.Current](#) is used.

lineWeight must be supplied if *layer* is supplied.

layer

- **Optional** [DwgLayer](#) object specifying the drawing layer to apply to the drawing circle.

When supplied, *layer* must be one of the following values:

- [DwgLAYER](#) object, representing a drawing layer existing in the opened drawing,
- [String](#) specifying the name of a drawing layer existing in the opened drawing.
- [DwgLayer.Current](#).

If *layer* is not supplied, [DwgLayer.Current](#) is used.

layer must be supplied if *lineType* is supplied.

lineType

- **Optional** [DwgLINETYPE](#) object specifying the drawing line type to apply to the new drawing circle.

lineType must be one of the following values:

- [DwgLINETYPE](#) object, representing a drawing line type existing in the opened drawing,
- [String](#) specifying the name of a drawing line type existing in the opened drawing,
- [DwgLineType.Current](#),
- [DwgLineType.ByLayer](#),
- [DwgLineType.ByBlock](#).

If *lineType* is not supplied, [DwgLineType.Current](#) is used.

drawing.createEllipse Method

```
createEllipse(  
    blockName,  
    DwgArc.ByCenterRadiusAngles,  
    center,  
    majorAxisEndPoint,  
    minor2majorRatio,  
    startAngle,  
    endAngle,  
    color,  
    lineWeight,  
    layer,  
    lineType  
)
```

Creates new drawing ellipse represented by [DwgELLIPSE](#) object.

Returns [DwgELLIPSE](#) object, representing the created drawing ellipse.

Parameters:

- blockName* - **Optional String** object specifying the name of a drawing block to which collection of the drawing objects the created drawing ellipse should be added.

When supplied, *blockName* must be one of the following:

- The name of a drawing block in a context of the [currently opened drawing](#). If the block with the specified name does not exist, it will be created with an empty collection of the drawing objects,
- [DwgBlock.OrphanBlock](#) to create the drawing ellipse which does not belong to the opened drawing (orphan). As an example, such ellipses are useful for on screen echoing when indicating an intention to have finally created drawing contents,
- [DwgBlock.Drawing](#) to add the created drawing ellipse to the end of the opened drawing body.

If *blockName* is not supplied, [DwgBlock.Drawing](#) is used.

- center* - [WPoint](#) object specifying location of the center of the drawing ellipse.

- majorAxisEndPoint* - [WPoint](#) object specifying location the endpoint of major axis of the drawing ellipse, relative to the center.

- minor2majorRatio* - **Number** specifying the ratio of minor axis to major axis of the drawing ellipse.

minor2majorRatio must be greater than zero (> 0.0) and less than or equal to 1.0.

- startAngle* - **Optional Number** specifying start angle of the drawing ellipse, supplied in radians.

If *startAngle* is not supplied, 0.0 is used.

startAngle must be supplied if *endAngle* is supplied.

- endAngle* - **Optional Number** specifying end angle of the drawing ellipse, supplied in radians.

If *endAngle* is not supplied, $2*\pi$ radians (360 degrees) is used.

endAngle must be supplied if *color* is supplied.

color

- **Optional** integer *Number* specifying the index of color to apply to the drawing ellipse.

color must be one of the following values:

- an integer value in range [1...255],
- `DwgColor.Current`,
- `DwgColor.Default`,
- `DwgColor.ByLayer`,
- `DwgColor.ByBlock`.

If *color* is not supplied, `DwgColor.Current` is used.

color must be supplied if *lineWeight* is supplied.

lineWeight

- **Optional** *Number* specifying the line weight to apply to the drawing ellipse.

When supplied, *lineWeight* must be one of the following values:

- a numeric value equal to or greater than zero (≥ 0.0),
- `DwgLineWeight.Current`,
- `DwgLineWeight.Default`,
- `DwgLineWeight.ByLayer`,
- `DwgLineWeight.ByBlock`.

If *lineWeight* is not supplied, `DwgLineWeight.Current` is used.

lineWeight must be supplied if *layer* is supplied.

layer

- **Optional** `DwgLayer` object specifying the drawing layer to apply to the drawing ellipse.

When supplied, *layer* must be one of the following values:

- `DwgLAYER` object, representing a drawing layer existing in the opened drawing,
- *String* specifying the name of a drawing layer existing in the opened drawing.
- `DwgLayer.Current`.

If *layer* is not supplied, `DwgLayer.Current` is used.

layer must be supplied if *lineType* is supplied.

lineType

- **Optional** `DwgLINETYPE` object specifying the drawing line type to apply to the new drawing ellipse.

lineType must be one of the following values:

- [DwgLINETYPE](#) object, representing a drawing line type existing in the opened drawing,
- [String](#) specifying the name of a drawing line type existing in the opened drawing,
- [DwgLineType.Current](#),
- [DwgLineType.ByLayer](#),
- [DwgLineType.ByBlock](#).

If *lineType* is not supplied, [DwgLineType.Current](#) is used.

drawing.createText Method

```
createText(  
    blockName,  
    location,  
    height,  
    widthFactor,  
    rotation,  
    obliqueAngle,  
    text,  
    backward,  
    upsideDown,  
    horzAlignment,  
    style,  
    color,  
    lineWeight,  
    layer,  
    lineType  
)
```

Creates new drawing text represented by [DwgTEXT](#) object.

Returns [DwgTEXT](#) object, representing the created drawing text.

Parameters:

- blockName* - **Optional String** object specifying the name of a drawing block to which collection of the drawing objects the created drawing text should be added.

When supplied, *blockName* must be one of the following:

- The name of a drawing block in a context of the [currently opened drawing](#). If the block with the specified name does not exist, it will be created with an empty collection of the drawing objects,

- `DwgBlock.OrphanBlock` to create the drawing text which does not belong to the opened drawing (orphan). As an example, such texts are useful for on screen echoing when indicating an intention to have finally created drawing contents,
- `DwgBlock.Drawing` to add the created drawing text to the end of the opened drawing body.

If *blockName* is not supplied, `DwgBlock.Drawing` is used.

- location* - `WPoint` object specifying location of the drawing text.
- height* - `Number` specifying the height of the drawing text. *height* must be greater than zero (>0.0).
- widthFactor* - **Optional** `Number` specifying the width factor of the drawing text.
- When supplied, *widthFactor* must be greater than zero (>0.0).
- If *widthFactor* is not supplied the value 1.0 is used.
- widthFactor* must be supplied when *rotation* is supplied.
- rotation* - **Optional** `Number` specifying the rotation angle of the drawing text, supplied in radians.
- If *rotation* is not supplied the value 0.0 is used.
- rotation* must be supplied when *obliqueAngle* is supplied.
- obliqueAngle* - **Optional** `Number` specifying the oblique angle of the drawing text, supplied in radians.
- When supplied, the normalized value of *obliqueAngle* must be in range corresponding to [-85, +85] degrees (+/-1.4835298641951 radians).
- If *obliqueAngle* is not supplied the value 0.0 is used.
- text* - `String` object specifying the text string applied to the drawing text and rendered on the screen when the text is visible.
- backward* - **Optional** `Number` specifying if the drawing text is backward (mirrored in X). If *backward* is `true`, the drawing text is mirrored in X.
- If *backward* is not supplied the value `false` is used.
- backward* must be supplied when *upsideDown* is supplied.
- upsideDown* - **Optional** `Number` specifying if the drawing text upside down (mirrored in Y). If *upsideDown* is `true`, the drawing text is mirrored in Y.
- If *upsideDown* is not supplied the value `false` is used.

textStyle

- **Optional** [DwgTEXTSTYLE](#) object specifying the drawing text style applied to the drawing text.

When supplied, *textStyle* must be one of the following values:

- [DwgTEXTSTYLE](#) object representing a text style, existing in the opened drawing,
- [String](#) object specifying the name of a text style, existing in the opened drawing,
- [DwgTextStyle.Current](#).

If *textStyle* is not supplied the value [DwgTextStyle.Current](#) is used.

color

- **Optional** integer [Number](#) specifying the index of color to apply to the drawing text.

color must be one of the following values:

- an integer value in range [1...255],
- [DwgColor.Current](#),
- [DwgColor.Default](#),
- [DwgColor.ByLayer](#),
- [DwgColor.ByBlock](#).

If *color* is not supplied, [DwgColor.Current](#) is used.

color must be supplied if *lineWeight* is supplied.

lineWeight

- **Optional** [Number](#) specifying the line weight to apply to the drawing text.

When supplied, *lineWeight* must be one of the following values:

- a numeric value equal to or greater than zero (≥ 0.0),
- [DwgLineWeight.Current](#),
- [DwgLineWeight.Default](#),
- [DwgLineWeight.ByLayer](#),
- [DwgLineWeight.ByBlock](#).

If *lineWeight* is not supplied, [DwgLineWeight.Current](#) is used.

lineWeight must be supplied if *layer* is supplied.

layer

- **Optional** [DwgLayer](#) object specifying the drawing layer to apply to the drawing text.

When supplied, *layer* must be one of the following values:

- [DwgLAYER](#) object, representing a drawing layer existing in the opened drawing,
- [String](#) specifying the name of a drawing layer existing in the opened drawing.
- [DwgLayer.Current](#).

If *layer* is not supplied, [DwgLayer.Current](#) is used.

layer must be supplied if *lineType* is supplied.

lineType

- **Optional** [DwgLINETYPE](#) object specifying the drawing line type to apply to the new drawing text.

lineType must be one of the following values:

- [DwgLINETYPE](#) object, representing a drawing line type existing in the opened drawing,
- [String](#) specifying the name of a drawing line type existing in the opened drawing,
- [DwgLineType.Current](#),
- [DwgLineType.ByLayer](#),
- [DwgLineType.ByBlock](#).

If *lineType* is not supplied, [DwgLineType.Current](#) is used.

drawing.redraw Method

redraw()

Redraws current view of the opened drawing on the screen.

Returns nothing.

Parameters: none.

drawing.getEntities Method

getEntities()

Creates an [Array](#) object consisting of [DwgObject](#) objects representing all drawing objects in the opened drawing body.

Returns [Array](#) of [DwgObject](#) objects.

Parameters: none

drawing.getSelectedEntities Method

getSelectedEntities()

Creates an [Array](#) object consisting of [DwgObject](#) objects representing all selected drawing objects in the opened drawing body.

Returns the created [Array](#) of [DwgObject](#) objects.

Parameters: none

drawing.setSelectedEntities Method

```
setSelectedEntities(  
    drawingObject1,  
    drawingObject2,  
    drawingObjectN,  
    drawingObjectArray1,  
    drawingObjectArray2,  
    drawingObjectArrayN  
)
```

Clears current selection of the drawing objects in the opened drawing body and, if specified, sets another selection of the drawing objects with the supplied ones.

Returns nothing.

Parameters:

Any number of objects of the following types in any combination:

- [DwgObject](#) representing a drawing object to be selected in the opened [drawing](#) body,

- [Array](#) object consisting of [DwgObjects](#) representing drawing object to be selected in the opened [drawing](#) body,
- `null` value to select nothing.

If no parameter is supplied no drawing objects are selected.

Empty [Arrays](#) are acceptable.

drawing.getLayers Method

getLayers()

Creates an [Array](#) object that consisting of [DwgLAYER](#) objects representing all drawing layers in the opened drawing.

Returns the created [Array](#) of [DwgLAYER](#) objects.

Parameters: none

drawing.getLayer Method

getLayer(`layerName`)

Returns [DwgLAYER](#) object representing the specified drawing layer in the opened drawing. If the layer does not exist, the returned value is `undefined`.

Parameters:

layerName - [String](#) object specifying the name of the drawing layer existing in the opened drawing. **Remark:** the layer names are case insensitive.

Example:

```
alert("Layer ", initDefPointsLayer().name, " is ready to use");

function initDefPointsLayer()
{
    var defpointsLayer = drawing.getLayer("DEFPOINTS");
}
```

```
if((defpointsLayer instanceof DwgLAYER) == false)
{
    // Create the layer if it does not exist
    defpointsLayer = drawing.createLayer("DEFPOINTS");
}

defpointsLayer.frozen = true;
defpointsLayer.lineWeight = DwgLineWeight.Default;
defpointsLayer.lineType = "CONTINUOUS";
defpointsLayer.color = DwgColor.Default;

return defpointsLayer;
}
```

drawing.createLayer Method

```
createLayer(
    layerName,
    frozen,
    locked,
    color,
    lineWeight,
    lineType,
)
```

Creates new drawing layer represented by [DwgLAYER](#) object.

Returns [DwgLAYER](#) object, representing the created drawing layer.

Parameters:

layerName - String object specifying the name of the drawing layer to create.

frozen - **Optional Boolean** value specifying if the drawing layer is frozen. The layer is frozen if *frozen* is true.

If *frozen* is not supplied, `false` is used.

frozen must be supplied if *locked* is supplied.

locked - **Optional Boolean** value specifying if the drawing layer is locked. The layer is locked if *locked* is true.

color - **Optional integer Number** specifying the index of color to apply to the drawing layer.

When supplied, *color* must be one of the following values:

- an integer value in range [1...255],
- `DwgColor.Default`.

If *color* is not supplied, `DwgColor.Default` is used.

color must be supplied if *lineWeight* is supplied.

lineWeight - **Optional** `Number` specifying the line weight to apply to the drawing layer.

When supplied, *lineWeight* must be one of the following values:

- a numeric value equal to or greater than zero (≥ 0.0),
- `DwgLineWeight.Default`.

If *lineWeight* is not supplied, `DwgLineWeight.Default` is used.

lineType - **Optional** `DwgLINETYPE` object specifying the drawing line type to apply to the drawing layer.

When supplied, *lineType* must be one of the following values:

- `DwgLINETYPE` object, representing a drawing line type existing in the opened drawing,
- `String` specifying the name of a drawing line type existing in the opened drawing.

If *lineType* is not supplied, "CONTINUOUS" line type is used.

drawing.getLineTypes Method

`getLineTypes()`

Creates an `Array` object consisting of `DwgLINETYPE` objects representing all drawing line types in the opened drawing.

Returns the created `Array` of `DwgLINETYPE` objects.

Parameters: none

drawing.getLineType Method

`getLineType(lineTypeName)`

Returns [DwgLINETYPE](#) object representing the specified drawing line type in the opened drawing. If the linetype does not exist, the returned value is `undefined`.

Parameters:

lineTypeName - *String* specifying the name of the drawing line type, existing in the opened drawing. *Remark:* the line type names are case insensitive.

drawing.getTextStyles Method

getTextStyles()

Creates an *Array* object that consisting of [DwgTEXTSTYLE](#) objects representing all drawing text styles in the opened drawing.

Returns the created *Array* of [DwgTEXTSTYLE](#) objects.

Parameters: none

drawing.getTextStyle Method

getTextStyle(textStyleName)

Returns [DwgTEXTSTYLE](#) object representing the specified drawing text style in the opened drawing. If the text style does not exist, the returned value is `undefined`.

Parameters:

textStyleName - *String* specifying the name of the drawing text style, existing in the opened drawing. *Remark:* the text style names are case insensitive.

Example:

```
if(checkMarkerTextStyle() == false)
{
    alert("To run this script, provide text style \"MARKER!\");
}

function checkMarkerTextStyle()
```

```

{
  var markerTextStyle = drawing.getTextStyle("MARKER");

  if((markerTextStyle instanceof DwgTEXTSTYLE) == false)
  {
    // the text style is not available - return false
    return false;
  }

  markerTextStyle.height = 5;
  markerTextStyle.obliqueAngle = 0;
  markerTextStyle.fontName = "ISOCP";
  markerTextStyle.widthFactor = 1;

  return true;
}

```



drawing.selectObject Method

selectObject(prompt)


Prompts user to select a drawing object available in the opened drawing.


Saves the input completion status in the property `drawing.selectObject.status`. If user completes input with a selected drawing object, the point on the opened drawing, at which user tapped the object, is attached as the `pickedAt`-property to the returned [DwgObject](#) representing the drawing object; the `pickedAt` is supplied as [WPoint](#) object.

The input completion status can be one of the following:

- `DwgResponse.Ok` if user selected a drawing object.
- `DwgResponse.DefaultEntered` if user selected no drawing object and clicked  or pressed Enter key.
- `DwgResponse.Cancel` if user cancelled input with clicked  or pressed `Escape` key.
- `DwgResponse.AbortRequested` if ShortCAD requested unconditional termination of the running script. For example, user may start another ShortCAD command having no completed or explicitly cancelled the running `ShortJS` script.

Returns one of the following values:

- [DwgObject](#) representing the drawing object, selected by user (if one),
- `DwgResponse.DefaultEntered` if user selected no object and clicked  or pressed Enter key.

- `DwgResponse.Cancel` if user cancelled input with clicked  or pressed `Escape` key.
- `DwgResponse.AbortRequested` if ShortCAD requested unconditional termination of the running script. For example, user may start another ShortCAD command having no completed or explicitly cancelled the running `ShortJS` script.

Parameters:

prompt - **Optional String** specifying the prompt which should be displayed in the prompt bar.

If *prompt* is not supplied the value "Select object? <done>" is used.

In case of using the Unicode characters for supplying *prompt*, the script must be saved with UTF-8 encoding.

Example:

```
for(;;) {
  var result = drawing.selectObject("Select an arc?:");
  if (result instanceof DwgARC) {
    alert("The including angle of arc is: ",
          result.endAngle - result.startAngle,
          "\nThe arc was selected at [",
          result.pickedAt.x, ", ", result.pickedAt.y, "]);
    continue;
  }
  switch (drawing.selectObject.status) {
    case DwgResponse.Ok:
    case DwgResponse.DefaultEntered:
      alert("Select a drawing arc!");
      continue;
    case DwgResponse.Cancel:
    case DwgResponse.AbortRequested:
      break;
  }
  break;
}
```




drawing.selectObjects Method

`selectObjects(prompt)`




Prompts user to select any number of drawing objects available in the opened drawing.

Saves the input completion status in the property `drawing.selectObjects.status`.

The input completion status can be one of the following:

- `DwgResponse.Ok` if user selected one or more drawing objects and clicked  or pressed Enter key to finish selecting the objects.
- `DwgResponse.DefaultEntered` if user selected no drawing objects and clicked  or pressed Enter key.
- `DwgResponse.Cancel` if user cancelled input having clicked  or pressed `Escape` key.
- `DwgResponse.AbortRequested` if ShortCAD requested unconditional termination of the running script. For example, user may start another ShortCAD command having no completed or explicitly cancelled the running `ShortJS` script.

Returns one of the following values:

- `Array` object consisting of `DwgObject` objects representing all the selected drawing objects, if user selected one or more drawing objects and clicked  or pressed `Enter` key to finish selecting the objects.
- `DwgResponse.DefaultEntered` if user selected no object and clicked  or pressed `Enter` key.
- `DwgResponse.Cancel` if user cancelled input having clicked  or pressed `Escape` key.
- `DwgResponse.AbortRequested` if ShortCAD requested unconditional termination of the running script. For example, user may start another ShortCAD command having no completed or explicitly cancelled the running `ShortJS` script.

Parameters:

prompt - **Optional String** specifying the prompt which should be displayed in the prompt bar.

If *prompt* is not supplied the value “Select objects? <done>” is used.

In case of using the Unicode characters for supplying *prompt*, the script must be saved with UTF-8 encoding.

Example:

```
for(;;) {
  result = drawing.selectObjects("Objects?");
  if (result instanceof Array) {
    alert(result.length,
      result.length == 1 ?
        " object was" : " objects were", " selected.");
    continue;
  }
  switch (drawing.selectObjects.status) {
    case DwgResponse.Ok:
    case DwgResponse.DefaultEntered:
```

```
        alert("Select drawing objects!");
        continue;
    case DwgResponse.Cancel:
    case DwgResponse.AbortRequested:
        break;
    }
    break;
}
```


drawing.getPoint Method


```
getPoint(
    basePoint,
    defaultPoint,
    promptBarPrompt,
    dialogBoxprompt,
    inputOption1,
    inputOption2,
    ...
    inputOptionN,
)
```

Prompts user to enter point on the opened drawing.



Saves the input completion status in the property `drawing.getPoint.status`. If a point is entered, the input completion status is also attached as `status`-property to the returned [WPoint](#) object representing the entered point.

The input completion status can be one of the following:

- `DwgResponse.Ok` if user entered a point on the opened drawing.
- `DwgResponse.CursorMoved` if yet, having no completed input, user moved the stylus or mouse cursor to a location on the opened drawing. `DwgResponse.CursorMoved` is noticed with every movement of the cursor. `DwgResponse.CursorMoved` is noticed only in case when one of the supplied (if any) `inputOptions` is `DwgGetPoint.Verbose` (see `inputOption` parameters below).
- `DwgResponse.IdleStateStarted` if yet, having no completed input, user moved the stylus or mouse cursor out of the viewed space of the opened drawing. This status is noticed only in case when one of the supplied (if any) `inputOptions` is `DwgGetPoint.Verbose` (see `inputOption` parameters below).
- `DwgResponse.DefaultEntered` if a default input was allowed and user clicked  or pressed Enter key.

- `DwgResponse.Cancel` if user cancelled input with clicked  or pressed `Escape` key.
- `DwgResponse.AbortRequested` if ShortCAD requested unconditional termination of the running script. For example, user may start another ShortCAD command having no completed or explicitly cancelled the running `ShortJS` script.

Returns one of the following values:

- `WPoint` object in one of the following cases:
 - Yet, having no completed input, the user moved the stylus or mouse cursor to a location on the opened drawing. In this case the value `DwgResponse.CursorMoved` is stored in `drawing.getPoint.status` and in the `status`-property, attached to the returned point.
 - The user completed input having entered a point. In this case the value `DwgResponse.Ok` is stored in `drawing.getPoint.status` and in the `status`-property, attached to the returned point.
 - The user completed input having entered the default value supplied in parameter `defaultPoint` (see below). In this case the value `DwgResponse.DefaultEntered` is stored in `drawing.getPoint.status` and in the `status`-property, attached to the returned point.
- `DwgResponse.IdleStateStarted` if yet, having no completed input, user moved the stylus or mouse cursor out of the viewed space of the opened drawing. This value returned only in case when `DwgGetPoint.Verbose` is supplied as an `inputOption` parameter (see below).
- `DwgResponse.DefaultEntered` if void default input was allowed and user clicked  or pressed `Enter` key. In this case the value `DwgResponse.DefaultEntered` is stored in `drawing.getPoint.status`.
- `DwgResponse.Cancel` if user cancelled input having clicked  or pressed `Escape` key.
- `DwgResponse.AbortRequested` if ShortCAD requested unconditional termination of the running script. For example, user may start another ShortCAD command having no completed or explicitly cancelled the running `ShortJS` script.


Parameters:

- basePoint*
- **Optional** `WPoint` object specifying a location to use it as base for relatively specified point. For example, entering major axis point of an ellipse may be based on the previously defined center of the ellipse, so the user could provide the required input with the relative or polar coordinates, added to the center location.

basePoint may be supplied as `undefined` value to disable a base location for entering point. Switching supplied *basePoint* from `WPoint` object to `undefined` value may help in conditional enabling of the base location. For example, when creating a sequence of linked vertexes in a loop, a prompt to enter the first

vertex can be supplied with `undefined` value for `basePoint` whereas the other vertexes could be entered each relatively to its predecessor.

defaultPoint


- **Optional** `WPoint` object specifying a location proposed as a default input. When supplied, the *defaultPoint* is returned in that case when the user completes input with clicked  or pressed `Enter` key.

defaultPoint may be supplied as `undefined` value to disable a default value for entering point. Switching supplied *defaultPoint* from `WPoint` object to `undefined` value may help in conditional enabling of the default input. For example, when creating a sequence of line segments in a loop, a prompt to enter the end points of the first two lines can be supplied with `undefined` value for *defaultPoint* whereas the end points of other lines could be supplied with the start point of the sequence to allow closing the sequence by default input.

- **Optional** `String` specifying the prompt which should be displayed in the prompt bar.

If *promptBarPrompt* is not supplied the string “Point?” is used.

In case of using the Unicode characters for supplying *promptBarPrompt*, the script must be saved with UTF-8 encoding.

- **Optional** `String` specifying the prompt which should be displayed in the dialog box, displayed when user taps .

If *dialogBoxprompt* is not supplied the string “Define Point” is used.

In case of using the Unicode characters for supplying *dialogBoxprompt*, the script must be saved with UTF-8 encoding.

inputOption1,
inputOption2,

...

inputOptionN

- **Optional** values specifying additional interactivity options.

When supplied, each *inputOption* must be one of the following:

- `DwgGetPoint.Verbose` to allow tracking the input completion status with any relevant event occurring with the stylus or mouse cursor. Such way, `DwgGetPoint.Verbose` option allows echoing the input of locations when creating or modifying drawing contents.

When `DwgGetPoint.Verbose` is supplied, `drawing.getPoint` returns with the first occurred cursor movement (see the statuses

`DwgResponse.CursorMoved` and `DwgResponse.IdleStateStarted` above).

When there is no `DwgGetPoint.Verbose` supplied, `drawing.getPoint` does not return until the user has completed or cancelled the input.

- `DwgGetPoint.AllowDefaultInput` to explicitly specify that the `WPoint` object, supplied by `defaultPoint`, must be used as a default input. `DwgGetPoint.AllowDefaultInput` makes sense only in case when `defaultPoint` is supplied but `basePoint` is not.
- `DwgGetPoint.AllowVoidDefaultInput` to explicitly specify that the default input is allowed, however no `WPoint` object will be returned as a default value. For example, `DwgGetPoint.AllowVoidDefaultInput` is used to finish scripts, fork application algorithms and break loops a reaction on the default user input.
- `DwgGetPoint.DisableDialogBox` to disable an option of using the ShortCAD point input dialog box.

`DwgGetPoint.DisableDialogBox` cannot be used together with `DwgGetPoint.StartWithDialogBox`.

- `DwgGetPoint.StartWithDialogBox` to start the entering a point with the ShortCAD point input dialog box.

`DwgGetPoint.StartWithDialogBox` cannot be used together with `DwgGetPoint.DisableDialogBox`.

- `DwgGetPoint.GetAngle` to configure the ShortCAD point input dialog box the way suitable of entering angles.

`DwgGetPoint.GetAngle` cannot be used together with `DwgGetPoint.GetDistance`.

- `DwgGetPoint.GetAngle` to configure the ShortCAD point input dialog box the way suitable of entering distances.

`DwgGetPoint.GetDistance` cannot be used together with `DwgGetPoint.GetAngle`.

- `DwgGetPoint.IgnoreOrthoSnap` to configure the ShortCAD point input dialog box the way suitable of entering distances.
- `DwgGetPoint.Nil` to specify no input option.

`DwgGetPoint.Nil` is useful for conditional switching any other input option. For example, using a variable `preferDialogInput` could switch the `DwgGetPoint.StartWithDialogBox` on or off:

```
getPoint("Center?", "Center of Spiral",
        preferDialogInput ?
            DwgGetPoint.StartWithDialogBox
            DwgGetPoint.Nil);
```

5. ShortCAD utility objects and functions

5.1. ShortCADInputForm Object

`ShortCADInputForm` object represents a dialog box, composed to collect user input by a number of data input fields. Any `ShortCADInputForm` consist of the form prompt and a number of the fields, each represented by its prompt and input control:

Input form prompt

Numeric field prompt:	2.5
Boolean field prompt:	Yes
String field prompt:	abc
Enum field prompt:	Choice 2
	Choice 1
	Choice 2
	Choice 3

The current version of ShortCAD Java script supports the input data types like:

- Number,
- Boolean,
- String.

Value of each input field in a created `ShortCADInputForm` is accessed via a unique assigned property, attached to the input form. For example, if an input form identified as `myForm` is collecting user input into the fields like layer name (string), layer visibility (boolean) and layer color (enum), then the form would have the attached properties like `layerName`, `layerVisibility` and `color` so the following expressions could implement the required application logic:

```
myForm = new ShortCADInputForm("New Layer definitions");
```

```
myForm.addInputField("layerName", // name of the property, attached to myForm
                    "Layer Name:",
                    "New Layer");

myForm.addInputField("layerVisibility", // name of the property, attached to
                    // myForm
                    "Layer is Visible:", true);

myForm.addInputField("colorIndex", // name of the property, attached to myForm
                    "Layer Color:",
                    1,
                    "Red\nYellow\nGreen\nCyan\nBlue\nMagenta\nWhite");

if (myForm.doModal() != false) {
    var name = myForm.layerName;
    var isLocked = (myForm.layerVisibility == false);
    var colorIndex = myForm.color;
}
```

ShortCADInputForm Object Properties

Properties of a [ShortCADInputForm](#) are defined programmatically by [ShortCADInputForm.addInputField](#) method.

Constructor of ShortCADInputForm Object

`new ShortCADInputForm(inputFormPrompt)`

Creates new [ShortCADInputForm](#) object with a prompt supplied by *inputFormPrompt*.

For example, a call like `new ShortCADInputForm("New Layer definitions")` will create an input form with the prompt displayed like:

New Layer definitions

ShortCADInputForm.addInputField Method

```
addInputField(
    attachedPropertyName,
    fieldPrompt,
    initialFieldValue,
    inputChoice
);
```

Adds new input field to the current [ShortCADInputForm](#) object.

Returned value is not defined.

Parameters:

attachedPropertyName - String specifying the name of the property, which is attached to the current [ShortCADInputForm](#) object to access the value, entered into the added field.

For example, call like `myForm.addInputField("layerName", "Layer Name:", "New Layer")` will attach property `layerName` to the input form `myForm` so the value entered into the field will be accessed by reference `myForm.layerName`.

fieldPrompt - String specifying the prompt displayed in the field.

For example, call like `myForm.addInputField("layerName", "Layer Name:", "New Layer")` will create a field with the prompt displayed like:



A screenshot of a text input field. The text 'Layer Name:' is displayed in a light blue font to the left of the input box. The input box contains the text 'New Layer'.

initialFieldValue - A value, initially assigned to the added input field. The type of the supplied *initialFieldValue* defines the type of data collected in the added input field. The type of data must be one of the following:

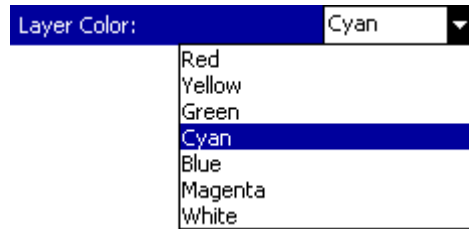
- Number,
- Boolean,
- String

inputChoice - **Optional** String supplying a list of possible values to enter into the added field.

When supplied, *inputChoice* is applied to the field only in case the type of initial value is **Numeric** integer value.

Items supplied in the *inputChoice* must be separated by the new line (`\n`) character. In this case the added field, will receive a flat enumerated integer value in range `[0...N]`, where `N` is the number of items supplied by *inputChoice*.

For example, call like `myForm.addInputField("layerName", "Layer Name:", "New Layer")` will create field with the prompt displayed like `myForm.addInputField("colorIndex", "Layer Color:", 3, "Red\nYellow\nGreen\nCyan\nBlue\nMagenta\nWhite")` will create a field with the dropdown list like:



ShortCADInputForm.doModal Method

doModal();

Launches the current [ShortCADInputForm](#) object to collect user input according to the preliminary [added input fields](#).


Returns `true` if the user completes the input with Ok. Otherwise, returns `false`.

5.2. ShortCADYield function

ShortCADYield()

Yields execution to ShortCAD for processing any pending UI or system events.

Returns one of the following values:

- `DwgResponse.Ok` nothing noticeable has occurred so the script may continue running.
- `DwgResponse.Cancel` if user clicked  or pressed `Escape` key.
- `DwgResponse.AbortRequested` if ShortCAD requested unconditional termination of the running script. For example, user may start another ShortCAD command having no completed or explicitly cancelled the running `ShortJS` script.

Remarks: `ShortCADYield()` is useful to interrupt longtime loops.


Parameters: none.

5.3. ShortCADSleep function

ShortCADSleep(milliseconds)

Suspends the execution of the current script until the time-out interval elapses.

Returns one of the following values:

- `DwgResponse.Ok` nothing noticeable has occurred so the script may continue running.
- `DwgResponse.Cancel` if user clicked  or pressed `Escape` key.
- `DwgResponse.AbortRequested` if ShortCAD requested unconditional termination of the running script. For example, user may start another ShortCAD command having no completed or explicitly cancelled the running `ShortJS` script.

Parameters:

milliseconds - `Number` specifying The time interval for which execution is to be suspended, in milliseconds.
