
After Effects Expression Reference

Release 0.0.3

Adobe Systems Incorporated

Feb 09, 2022

INTRODUCTION

1	Introduction	3
2	Changelog	5
2.1	After Effects 17.7 (Feb 2021)	5
2.2	After Effects 17.6 (Jan 2021)	5
2.3	After Effects 17.1.2 (Jul 2020)	5
2.4	After Effects 17.1 (May 19 2020)	5
2.5	After Effects 17.0.5 (Mar 2020)	6
2.6	After Effects 17.0.2 (Jan 2020)	6
2.7	After Effects 17.0 (Jan 24 2020)	6
2.8	After Effects 16.1.3 (Sep 2019)	7
2.9	After Effects 16.1.2 (June 2019)	7
2.10	After Effects 16.1 (CC 19) (Apr 2 2019)	7
2.11	After Effects 16.0 (CC 19) (Oct 15 2018)	8
2.12	After Effects 15.1.2 (Jul 16 2018)	8
2.13	After Effects 15.1 (Apr 3 2018)	8
2.14	After Effects 15.0 (CC) (Oct 18 2017)	8
2.15	After Effects 13.6 (CC 2015) (Nov 30 2015)	9
2.16	After Effects 13.5 (CC 2015) (Jun 15 2015)	9
2.17	After Effects 13.2 (CC 2014.2) (Dec 16 2014)	9
2.18	After Effects 12.1 (CC) (Sep 8 2013)	9
2.19	After Effects 10.5 (CS5.5) (Apr 11 2011)	10
2.20	After Effects 5.5 (Jan 7 2002)	10
2.21	After Effects 5.0 (Apr 2001)	10
3	Resources	11
4	Examples	13
4.1	Get this project's AEP name (AE 15.1+ only)	13
4.2	Make a layer revolve in a circle	13
4.3	Rotate the hands of a clock	14
4.4	Position one layer between two others	14
4.5	Create a trail of images	14
4.6	Create a bulge between two layers	15
4.7	Fade opacity of a 3D layer based on distance from camera	15
4.8	Make a 3D layer invisible if facing away from camera	16
4.9	Flip layer horizontally if facing away from camera	16
4.10	Animate scale at each layer marker	16
4.11	Start or stop wiggle at specific time	17
4.12	Match camera focal plane to another layer	18

5	Global	19
5.1	comp(name)	19
5.2	footage(name)	19
5.3	thisProject	20
5.4	thisComp	20
5.5	thisLayer	20
5.6	thisProperty	20
5.7	time	21
5.8	colorDepth	21
5.9	posterizeTime(updatesPerSecond)	21
5.10	value	22
6	Time Conversion	23
6.1	timeToFrames(t=time + thisComp.displayStartTime, fps=1.0 / thisComp.frameDuration, isDuration=false)	23
6.2	framesToTime(frames, fps=1.0 / thisComp.frameDuration)	24
6.3	timeToTimecode(t=time + thisComp.displayStartTime, timecodeBase=30, isDuration=false)	24
6.4	timeToNTSCTimecode(t=time + thisComp.displayStartTime, ntscDropFrame=false, isDuration=false)	24
6.5	timeToFeetAndFrames(t=time + thisComp.displayStartTime, fps=1.0 / thisComp.frameDuration, framesPerFoot=16, isDuration=false)	25
6.6	timeToCurrentFormat(t=time + thisComp.displayStartTime, fps=1.0 / thisComp.frameDuration, isDuration=false)	25
7	Vector Math	27
7.1	add(vec1, vec2)	27
7.2	sub(vec1, vec2)	27
7.3	mul(vec, amount)	28
7.4	div(vec, amount)	28
7.5	clamp(value, limit1, limit2)	28
7.6	dot(vec1, vec2)	29
7.7	cross(vec1, vec2)	29
7.8	normalize(vec)	29
7.9	length(vec)	30
7.10	length(point1, point2)	30
7.11	lookAt(fromPoint, atPoint)	30
8	Random Numbers	33
8.1	seedRandom(offset, timeless=false)	33
8.2	random()	34
8.3	random(maxValOrArray)	34
8.4	random(minValOrArray, maxValOrArray)	34
8.5	gaussRandom()	35
8.6	gaussRandom(maxValOrArray)	35
8.7	gaussRandom(minValOrArray, maxValOrArray)	35
8.8	noise(valOrArray)	36
9	Interpolation	37
9.1	linear(t, tMin, tMax, value1, value2)	37
9.2	linear(t, value1, value2)	38
9.3	ease(t, tMin, tMax, value1, value2)	38
9.4	ease(t, value1, value2)	39
9.5	easeIn(t, tMin, tMax, value1, value2)	39
9.6	easeIn(t, value1, value2)	39

9.7	<code>easeOut(t, tMin, tMax, value1, value2)</code>	40
9.8	<code>easeOut(t, value1, value2)</code>	40
10	Color Conversion	41
10.1	<code>rgbToHsl(rgbaArray)</code>	41
10.2	<code>hslToRgb(hslaArray)</code>	41
10.3	<code>hexToRgb(hexString)</code>	42
11	Other Math	43
11.1	<code>degreesToRadians(degrees)</code>	43
11.2	<code>radiansToDegrees(radians)</code>	43
12	Project	45
12.1	<code>Project.fullPath</code>	45
12.2	<code>Project.bitsPerChannel</code>	45
12.3	<code>Project.linearBlending</code>	46
13	Comp	47
13.1	<code>Comp.layer(index)</code>	47
13.2	<code>Comp.layer(name)</code>	47
13.3	<code>Comp.layer(otherLayer, relIndex)</code>	48
13.4	<code>Comp.layerByComment(comment)</code>	48
13.5	<code>Comp.marker</code>	48
13.6	<code>Comp.marker.key(index)</code>	49
13.7	<code>Comp.marker.key(name)</code>	49
13.8	<code>Comp.marker.nearestKey(t)</code>	50
13.9	<code>Comp.marker.numKeys</code>	50
13.10	<code>Comp.numLayers</code>	50
13.11	<code>Comp.activeCamera</code>	51
13.12	<code>Comp.width</code>	51
13.13	<code>Comp.height</code>	51
13.14	<code>Comp.duration</code>	51
13.15	<code>Comp.ntscDropFrame</code>	52
13.16	<code>Comp.displayStartTime</code>	52
13.17	<code>Comp.frameDuration</code>	52
13.18	<code>Comp.shutterAngle</code>	52
13.19	<code>Comp.shutterPhase</code>	53
13.20	<code>Comp.bgColor</code>	53
13.21	<code>Comp.pixelAspect</code>	53
13.22	<code>Comp.name</code>	53
14	Footage	55
14.1	<code>Footage.width</code>	55
14.2	<code>Footage.height</code>	55
14.3	<code>Footage.duration</code>	55
14.4	<code>Footage.frameDuration</code>	56
14.5	<code>Footage.ntscDropFrame</code>	56
14.6	<code>Footage.pixelAspect</code>	56
14.7	<code>Footage.name</code>	56
14.8	<code>Footage.sourceText</code>	57
14.9	<code>Footage.sourceData</code>	57
14.10	<code>Footage.dataValue(dataPath)</code>	58
14.11	<code>Footage.dataKeyCount(dataPath)</code>	58
14.12	<code>Footage.dataKeyTimes(dataPath, t0 = startTime, t1 = endTime)</code>	59
14.13	<code>Footage.dataKeyValues(dataPath, t0 = startTime, t1 = endTime)</code>	59

15 Camera	61
15.1 Camera.pointOfInterest	61
15.2 Camera.zoom	62
15.3 Camera.depthOfField	62
15.4 Camera.focusDistance	62
15.5 Camera.aperture	62
15.6 Camera.blurLevel	63
15.7 Camera.irisShape	63
15.8 Camera.irisRotation	63
15.9 Camera.irisRoundness	63
15.10 Camera.irisAspectRatio	64
15.11 Camera.irisDiffractionFringe	64
15.12 Camera.highlightGain	64
15.13 Camera.highlightThreshold	64
15.14 Camera.highlightSaturation	65
15.15 Camera.active	65
16 Light	67
16.1 Light.pointOfInterest	67
16.2 Light.intensity	68
16.3 Light.color	68
16.4 Light.coneAngle	68
16.5 Light.coneFeather	68
16.6 Light.shadowDarkness	69
16.7 Light.shadowDiffusion	69
17 Group	71
17.1 numProperties	72
17.2 propertyGroup(countUp=1)	72
17.3 propertyIndex	72
17.4 name	72
18 Effect	73
18.1 Effect.active	73
18.2 Effect.param(name)	73
18.3 Effect.param(index)	73
19 Mask	75
19.1 Mask.maskOpacity	75
19.2 Mask.maskFeather	75
19.3 Mask.maskExpansion	75
19.4 Mask.invert	76
20 Property	77
20.1 value	77
20.2 valueAtTime(t)	77
20.3 velocity	78
20.4 velocityAtTime(t)	78
20.5 speed	78
20.6 speedAtTime(t)	79
20.7 wiggle(freq, amp, octaves=1, amp_mult=0.5, t=time)	79
20.8 temporalWiggle(freq, amp, octaves=1, amp_mult=0.5, t=time)	80
20.9 smooth(width=.2, samples=5, t=time)	81
20.10 loopIn(type="cycle", numKeyframes=0)	81
20.11 loopOut(type="cycle", numKeyframes=0)	82

20.12	loopInDuration(type="cycle", duration=0)	82
20.13	loopOutDuration(type="cycle", duration=0)	83
20.14	key(index)	83
20.15	key(markerName)	83
20.16	nearestKey(t)	84
20.17	numKeys	84
20.18	propertyGroup(countUp=1)	84
20.19	propertyIndex	85
20.20	name	85
21	Path Property	87
21.1	name	88
21.2	PathProperty.points(t=time)	88
21.3	PathProperty.inTangents(t=time)	88
21.4	PathProperty.outTangents(t=time)	89
21.5	PathProperty.isClosed()	89
21.6	PathProperty.pointOnPath(percentage=0.5, t=time)	89
21.7	PathProperty.tangentOnPath(percentage=0.5, t=time)	90
21.8	PathProperty.normalOnPath(percentage=0.5, t=time)	90
21.9	PathProperty.createPath(points=[[0,0], [100,0], [100,100], [0,100]], inTangents=[], outTangents=[], is_closed=true)	91
22	Key	93
22.1	Key.value	93
22.2	Key.time	93
22.3	Key.index	94
23	MarkerKey	95
23.1	Marker.duration	96
23.2	Marker.comment	96
23.3	Marker.chapter	96
23.4	Marker.url	96
23.5	Marker.frameTarget	97
23.6	Marker.eventCuePoint	97
23.7	Marker.cuePointName	97
23.8	Marker.parameters	97
23.9	Marker.protectedRegion	98
24	Layer Sub-objects	99
24.1	Layer.source	99
24.2	Layer.sourceTime(t=time)	99
24.3	Layer.sourceRectAtTime(t = time, includeExtents = false)	100
24.4	Layer.effect(name)	100
24.5	Layer.effect(index)	101
24.6	Layer.mask(name)	101
24.7	Layer.mask(index)	101
25	Layer General	103
25.1	Layer.width	103
25.2	Layer.height	103
25.3	Layer.index	103
25.4	Layer.parent	104
25.5	Layer.hasParent	104
25.6	Layer.inPoint	104
25.7	Layer.outPoint	105

25.8	Layer.startTime	105
25.9	Layer.hasVideo	105
25.10	Layer.hasAudio	105
25.11	Layer.active	106
25.12	Layer.enabled	106
25.13	Layer.audioActive	106
25.14	Layer.sampleImage(point, radius=[0.5, 0.5], postEffect=true, t=time)	106
26	Layer Properties	109
26.1	Layer.anchorPoint	109
26.2	Layer.position	109
26.3	Layer.scale	110
26.4	Layer.rotation	110
26.5	Layer.opacity	110
26.6	Layer.audioLevels	110
26.7	Layer.timeRemap	111
26.8	Layer.marker.key(index)	111
26.9	Layer.marker.key(name)	111
26.10	Layer.marker.nearestKey(t)	112
26.11	Layer.marker.numKeys	112
26.12	Layer.name	112
27	Layer Space Transforms	113
27.1	toComp(point, t=time)	113
27.2	fromComp(point, t=time)	114
27.3	toWorld(point, t=time)	114
27.4	fromWorld(point, t=time)	115
27.5	toCompVec(vec, t=time)	115
27.6	fromCompVec(vec, t=time)	115
27.7	toWorldVec(vec, t=time)	116
27.8	fromWorldVec(vec, t=time)	116
27.9	fromCompToSurface(point, t=time)	117
28	Layer 3D	119
28.1	Layer.orientation	119
28.2	Layer.rotationX	119
28.3	Layer.rotationY	119
28.4	Layer.rotationZ	120
28.5	Layer.lightTransmission	120
28.6	Layer.castsShadows	120
28.7	Layer.acceptsShadows	120
28.8	Layer.acceptsLights	121
28.9	Layer.ambient	121
28.10	Layer.diffuse	121
28.11	Layer.specular	121
28.12	Layer.shininess	122
28.13	Layer.metal	122
29	Text	123
29.1	Text.sourceText	123
29.2	Text.Font...	123
30	Source Text	125
30.1	SourceText.style	125
30.2	SourceText.getStyleAt(charIndex, t = time)	125

30.3	SourceText.createStyle()	126
31	Text Style	127
31.1	TextStyle.setText(value)	127
31.2	TextStyle.fontSize	128
31.3	TextStyle.setFontSize(value)	128
31.4	TextStyle.font	128
31.5	TextStyle.setFont(value)	128
31.6	TextStyle.isFauxBold	129
31.7	TextStyle.setFauxBold(value)	129
31.8	TextStyle.isFauxItalic	129
31.9	TextStyle.setFauxItalic(value)	129
31.10	TextStyle.isAllCaps	130
31.11	TextStyle.setAllCaps(value)	130
31.12	TextStyle.isSmallCaps	130
31.13	TextStyle.setSmallCaps(value)	130
31.14	TextStyle.tracking	131
31.15	TextStyle.setTracking(value)	131
31.16	TextStyle.leading	131
31.17	TextStyle.setLeading(value)	131
31.18	TextStyle.isAutoLeading	132
31.19	TextStyle.setAutoLeading(value)	132
31.20	TextStyle.baselineShift	132
31.21	TextStyle.setBaselineShift(value)	132
31.22	TextStyle.applyFill	133
31.23	TextStyle.setApplyFill(value)	133
31.24	TextStyle.fillColor	133
31.25	TextStyle.setFill-color(value)	133
31.26	TextStyle.applyStroke	134
31.27	TextStyle.setApplyStroke(value)	134
31.28	TextStyle.strokeColor	134
31.29	TextStyle.setStrokeColor(value)	134
31.30	TextStyle.strokeWidth	135
31.31	TextStyle.setStrokeWidth(value)	135

The reference is still being edited and improved upon.

INTRODUCTION

Use the After Effects expression elements along with standard JavaScript elements to write your expressions. You can use the Expression Language menu at any time to insert methods and attributes into an expression, and you can use the pick whip at any time to insert properties.

If an argument description contains an equal sign (=) and a value (such as `t=time` or `width=.2`), then the argument uses the included default value if you don't specify a different value.

Some argument descriptions include a number in square brackets—this number indicates the dimension of the expected property or Array.

Some return-value descriptions include a number in square brackets—this number specifies the dimension of the returned property or Array. If a specific dimension is not included, the dimension of the returned Array depends on the dimension of the input.

The W3Schools JavaScript reference website provides information for the standard JavaScript language, including pages for the JavaScript Math and String objects.

CHANGELOG

What's new and changed for expressions?

2.1 After Effects 17.7 (Feb 2021)

- Fixed: An issue where expression edits made in the Graph Editor were not applied consistently.
-

2.2 After Effects 17.6 (Jan 2021)

- Fixed: An issue that could cause an expression to be replaced instead of appending when using expression or property pick-whip.
-

2.3 After Effects 17.1.2 (Jul 2020)

- Fixed: An issue where Markers could not be referenced by name in the JavaScript Expressions Engine.
-

2.4 After Effects 17.1 (May 19 2020)

- Fixed: An issue with Expression editor to auto-complete 'timeToFrames' function.
-

2.5 After Effects 17.0.5 (Mar 2020)

- Fixed: An issue where the Link Focus to Layer command produced an expression that did not work with the JavaScript expression engine.
-

2.6 After Effects 17.0.2 (Jan 2020)

- Fixed: An issue where wrong line numbers would be displayed related to errors in JavaScript expressions.
-

2.7 After Effects 17.0 (Jan 24 2020)

- Implemented Dropdown Menu Expression Control
 - Expression Editor improvements:
 - You can now use the new scrolling functionality to prevent the scroll from adjusting incorrectly when the box is resized by typing the return character.
 - Prevent numbers from matching in an autocomplete list if the variable begins with a number. Smarter autocomplete prevents from overriding closing brackets and quotes.
 - You can now scale font size for Hi-DPI displays.
 - Graph editor now commits changes in preferences for all the open graph editors.
 - If you enable syntax highlight, the folding icon buttons in the UI now respect the default and background color, or the line numbers color and background color.
 - Expression performance improvements:
 - After Effects now attempts to detect an expression that does not change throughout a comp and calculates the expression only once. Load your favorite expression-filled comp and experience the improved performance.
 - Any expression using *posterizeTime()* now calculates only once for the entire comp, not on every frame.
 - Added: Extended expressions access to Text properties.
 - Added: *Text.Font...*
 - Added: *Source Text*
 - Added: *Text Style*
-

2.8 After Effects 16.1.3 (Sep 2019)

- Fixed: Indentation of curly braces on new lines could be incorrect in the Expressions editor.
-

2.9 After Effects 16.1.2 (June 2019)

- Fixed: After Effects crashes when you close a project that has an expression containing an error.
 - Fixed: Expression error messages could be truncated in the error ribbon if there were multiple lines of error text to show.
 - Fixed: The property, `this_Layer` had stopped working when using the Legacy ExtendScript expression engine.
 - Fixed: Crash when switching the project level expression engine from JavaScript to Legacy ExtendScript.
 - Fixed: Crash with expressions that contain calls to `Date.toLocaleString()`.
 - Fixed: Crash when editing expressions in the Graph Editor expression field when AutoComplete is disabled.
-

2.10 After Effects 16.1 (CC 19) (Apr 2 2019)

- Implemented new expression editor
 - Fixed: The JavaScript expressions engine does not generate the same random number results as the Legacy ExtendScript engine.
 - Fixed: When an expression references the name of a layer in a string or in a Source Text property, the name of the layer is not returned. Instead, it returns `[Object]`.
 - Fixed: The `sampleImage()` expression method returns the wrong value if the post-expression value of the property is read by a ScriptUI panel.
 - Fixed: Applying the `createPath()` expression via the Expression Language menu auto-fills the `(is_Closed)` parameter as deprecated snake case instead of camel case `isClosed`.
 - Fixed: Renaming an effect that is referenced by an expression causes the expression to incorrectly update references to that effect properties when those properties have the same name as the effect.
 - Fixed: The Link Focus Distance to Layer, Link Focus Distance to Point of Interest, Create Stereo 3D Rig, and Create Orbit Null commands create expressions that are incompatible with the JavaScript expression engine.
 - Fixed: Specific complex, multi-composition expressions cause fast flickering of the expression error warning banner and icons. Note that to fix this, there is a small slowdown in expression evaluation speed for these expressions.
-

2.11 After Effects 16.0 (CC 19) (Oct 15 2018)

- Implemented new Javascript engine
 - Added: *hexToRgb*
 - Added: *marker protectedRegion* property
-

2.12 After Effects 15.1.2 (Jul 16 2018)

- Fixed: If your project contains multiple master properties by the same name, the expressions that refer to the master properties evaluate incorrectly.
 - Fixed: The Property Link pick whip incorrectly writes a self-referential expression for the other selected properties.
-

2.13 After Effects 15.1 (Apr 3 2018)

- Added: Property Link pick whip
 - Added: Support for custom expression function libraries
 - Added: Expression access to *Project*
 - Added: *Project.fullPath*
 - Added: *Project.bitsPerChannel*
 - Added: *Project.linearBlending*
-

2.14 After Effects 15.0 (CC) (Oct 18 2017)

- Added: Expression access to data in JSON files
 - Added: *footage sourceText* attribute
 - Added: *footage sourceData* attribute
 - Added: *footage dataValue* method
 - Added: *footage dataKeyCount* method
 - Added: *footage dataKeyTimes* method
 - Added: *footage dataKeyValues* method
- Added: Expression access to path points on masks, Bezier shapes, and brush strokes
 - Added: *path points* method
 - Added: *path inTangents* method

- Added: *path outTangents* method
 - Added: *path isClosed* method
 - Added: *path pointOnPath* method
 - Added: *path tangentOnPath* method
 - Added: *path normalOnPath* method
 - Added: *path createPath* method
-

2.15 After Effects 13.6 (CC 2015) (Nov 30 2015)

- Improved performance of expressions on time-remapped layers. This also reduces rendering time for audio on time-remapped layers with expressions.
 - Fixed: Changing the source text of a text layer no longer causes expressions to fail when the name of the text layer was referenced.
 - Fixed: After Effects no longer crashes when the graph editor is displayed while processing an time remapping expression.
-

2.16 After Effects 13.5 (CC 2015) (Jun 15 2015)

- More efficient expression evaluation
 - Added: Expression warning banner
-

2.17 After Effects 13.2 (CC 2014.2) (Dec 16 2014)

- Added: *sourceRectAtTime()* method
 - Fixed: *sampleImage()* in an expression no longer disables multiprocessing
-

2.18 After Effects 12.1 (CC) (Sep 8 2013)

- Added iris and highlight properties for camera layers to the expression language menu
 - Added: *Camera.irisShape*
 - Added: *Camera.irisRotation*
 - Added: *Camera.irisRoundness*
 - Added: *Camera.irisAspectRatio*
 - Added: *Camera.irisDiffractionFringe*
-

- Added: *Camera.highlightGain*
 - Added: *Camera.highlightThreshold*
 - Added: *Camera.highlightSaturation*
-

2.19 After Effects 10.5 (CS5.5) (Apr 11 2011)

- Added: *Footage.ntscDropFrame*
 - Added: *ntscDropFrame* argument to *timeToCurrentFormat()*
 - Added: *Layer.sourceTime()*
-

2.20 After Effects 5.5 (Jan 7 2002)

- Added: Looping via expressions
 - Added: Expression controllers
-

2.21 After Effects 5.0 (Apr 2001)

- Expressions first added

RESOURCES

The user ‘Beaver’ posted [5 Expressions that will change your life](#) on the Mograph forums.

Dan Ebberts provides example expressions and tutorials for learning how to work with expressions on his [MotionScript website](#). For example, Dan provides an excellent [page about collision detection](#).

Colin Braley provides a tutorial and example project on [their site](#) that show how to use expressions to make one layer repel others in a natural-seeming manner.

The AE Enhancers forum provides many examples and much useful information about expressions, as well as scripts and animation presets. In [this post on the AE Enhancers forum](#), Paul Tuersley provides a tutorial and example project that show how to use expressions to animate several layers in a swarm.

Rick Gerard provides an example on [Rick’s site](#) that demonstrates rolling a square object along a floor so that the sides stay in contact with the floor plane.

Carl Larsen provides a video tutorial on the [Creative COW website](#) that demonstrates how to use expressions and parenting to relate the rotation of a set of wheels to the horizontal movement of a vehicle.

Chris Zwar provides an example project on [chriszwar.com](#) for automatically arranging still images or videos into a grid (like a video wall). You can easily adjust position and spacing with sliders that are connected to a system of expressions. There are three compositions in the project—one for stills, one for videos, and one to create an auto-storyboard in which a video is sampled at user-defined intervals and aligned into a grid.

[JJ Gifford’s website](#) provides several example projects that demonstrate how to use expressions.

Maltaannon (Jerzy Drozda, Jr.) provides a video tutorial on [maltaanon.com](#) that shows how to use expressions to create a volume meter using the results of the Convert Audio To Keyframes command.

EXAMPLES

Note: Many of the examples in this section are based on expressions provided by Dan Ebberts.

4.1 Get this project's AEP name (AE 15.1+ only)

While there is no method to directly access your AEP's name, you CAN get the full path to the AEP.

With some string manipulation, you can derive the aep name from this:

```
var aepName = thisProject.fullPath.split($.os.indexOf("Windows") > -1 ? "\\\" : "/").  
↳pop();
```

If you wanted to write “Unsaved” in that case, you can use the following expression:

```
var aepName = thisProject.fullPath.split($.os.indexOf("Windows") > -1 ? "\\\" : "/").  
↳pop();  
aepName = aepName === "" ? "Unsaved" : aepName;
```

4.2 Make a layer revolve in a circle

You can create an expression without using properties from other layers. For example, you can make a layer revolve in a perfect circle.

1. Select a layer, press P to reveal its Position property in the Timeline panel, and Alt-click (Windows) or Option-click (Mac OS) the stopwatch to the left of the property name.
2. Enter the following in the expression field:

```
[(thisComp.width/2), (thisComp.height/2)] + [Math.sin(time)*50, -Math.cos(time)*50]
```

4.3 Rotate the hands of a clock

You can use the pick whip to link rotation values between layers to animate the hands on a clock—as the hour hand moves from hour to hour, the minute hand rotates the full circumference of the clock face. This type of animation would take a long time to create if you had to set each keyframe for both hand layers, but with the pick whip, you can do it in a matter of minutes.

1. Import or create two long, narrow solid-color layers: an hour hand and a minute hand.
2. Set the anchor points at the ends of the layers.
3. Move the layers so that the anchor points are at the center of the composition.
4. Set Rotation keyframes for the hour hand.
5. Select the Rotation property for the minute hand and choose **Animation > Add Expression**.
6. Drag the pick whip to the Rotation property for the hour hand. The following expression appears:

```
thisComp.layer("hour hand").rotation
```

7. To make the minute hand rotate 12 times as fast as the hour hand, add `* 12` at the end of the expression as follows:

```
thisComp.layer("hour hand").rotation * 12
```

4.4 Position one layer between two others

This example expression positions and maintains one layer at a balanced distance between two other layers.

1. Start with three layers.
2. Animate the positions of the first two layers in the Timeline panel.
3. Select the third layer, press P to reveal the Position property, and Alt-click (Windows) or Option-click (Mac OS) the stopwatch button to the left of the property name.
4. Enter the following in the expression field:

```
(thisComp.layer(1).position + thisComp.layer(2).position)/2
```

4.5 Create a trail of images

This example expression instructs a layer to be at the same position as the next higher layer in the Timeline panel, but delayed by a specified amount of time (in this case, 0.5 seconds). You can set similar expressions for the other geometric properties.

1. Start with two solid-color layers that are scaled to approximately 30% of the composition size. (See Solid-color layers and solid-color footage items.)
2. Animate the position of the first layer.

3. Select the second layer, press P to reveal the Position property, and Alt-click (Windows) or Option-click (Mac OS) the stopwatch button to the left of the property name.
4. Enter the following in the expression field:

```
thisComp.layer(thisLayer, -1).position.valueAtTime(time - .5)
```

5. Duplicate the last layer five times by selecting it and pressing Ctrl+D (Windows) or Command+D (Mac OS) five times.

All layers follow the same path, and each is delayed 0.5 seconds from the previous.

Note: Dan Ebberts provides more examples and techniques for creating trails of images on his [MotionScript](#).

4.6 Create a bulge between two layers

This example expression synchronizes the Bulge Center argument of the Bulge effect in one layer with the position of another layer. For example, you can create an effect that looks like a magnifying glass moving over a layer, with the contents under the magnifying glass bulging as the lens (that is, the overlying layer) moves. This expression uses the `fromWorld` method, which makes the expression work correctly regardless of whether you move the magnifying glass layer or the underlying layer. You can rotate or scale the underlying layer, and the expression stays intact.

You can also use other effects, such as Ripple, with this expression.

1. Start with two layers. Make one layer a magnifying glass or similar object with a hole in the middle and name it Magnifier. (See [Creating layers](#).)
2. Animate the position of the magnifying glass layer. (See [Motion paths](#).)
3. Apply the Bulge effect to the other layer. (See [Apply an effect or animation preset](#).)
4. Select the Bulge Center property of the Bulge effect in the Timeline panel and choose Animation > Add Expression, or Alt-click (Windows) or Option-click (Mac OS) the stopwatch button for the property.
5. Select the default expression text and type the following:

```
fromWorld(thisComp.layer("Magnifier").position)
```

4.7 Fade opacity of a 3D layer based on distance from camera

Apply the following expression to the Opacity property of a 3D layer:

```
startFade = 500; // Start fade 500 pixels from camera.
endFade = 1500; // End fade 1500 pixels from camera.

try { // Check whether there's a camera
C = thisComp.activeCamera.toWorld([0,0,0]);
} catch (err) { // No camera, so assume 50mm
w = thisComp.width * thisComp.pixelAspect;
```

(continues on next page)

(continued from previous page)

```
z = (w/2)/Math.tan(degreesToRadians(19.799));  
C = [0,0,-z];  
}  
  
P = toWorld(anchorPoint);  
d = length(C,P);  
  
linear(d,startFade,endFade,100,0)
```

The fade starts at a distance of 500 pixels from the camera and is complete at 1500 pixels from the camera. The linear interpolation method is used to map distance values to opacity values.

4.8 Make a 3D layer invisible if facing away from camera

Apply the following expression to the Opacity property of a 3D layer:

```
if (toCompVec([0, 0, 1])[2] > 0 ) value else 0
```

Note: Dan Ebberts explains this expression on his [site](#).

4.9 Flip layer horizontally if facing away from camera

Apply the following expression to the Scale property of a 3D layer:

```
if (toCompVec([0, 0, 1])[2] > 0 ) value else [-value[0], value[1], value[2]]
```

4.10 Animate scale at each layer marker

Apply the following expression to a Scale property to make a layer wobble at each marker:

```
n = 0;  
t = 0;  
  
if (marker.numKeys > 0){  
  n = marker.nearestKey(time).index;  
  if (marker.key(n).time > time) n--;  
}  
  
if (n > 0) t = time - marker.key(n).time;
```

(continues on next page)

(continued from previous page)

```
amp = 15;
freq = 5;
decay = 3.0;

angle = freq * 2 * Math.PI * t;
scaleFact = (100 + amp * Math.sin(angle) / Math.exp(decay * t)) / 100;
[value[0] * scaleFact, value[1] / scaleFact];
```

4.11 Start or stop wiggle at specific time

You can use any expression in place of the wiggle expression used here, to begin and end the influence of any expression at a specific time.

Apply the following expression to a property to wiggle it beginning at time 2 seconds:

```
timeToStart = 2;
if (time > timeToStart) {
  wiggle(3,25);
} else {
  value;
}
```

Apply the following expression to a property to stop wiggling it at time 4 seconds:

```
timeToStop = 4;

if (time > timeToStop) {
  value;
} else {
  wiggle(3,25);
}
```

Apply the following expression to a property to start wiggling it at time 2 seconds and stop wiggling it at time 4 seconds:

```
timeToStart = 2;
timeToStop = 4;

if ((time > timeToStart) && (time < timeToStop)) {
  wiggle(3,25);
} else {
  value;
}
```

4.12 Match camera focal plane to another layer

Apply the following expression to the Focus Distance property of a camera layer to have its focus distance match the distance to the anchor point of a layer named “target”:

```
target = thisComp.layer("target");
V1 = target.toWorld(target.anchorPoint) - toWorld([0,0,0]);
V2 = toWorldVec([0,0,1]);
dot(V1,V2);
```

Note: Dan Ebberts explains this expression example in detail on his [website](#).

5.1 comp(name)

Description

Retrieves another composition by name.

Parameters

name	String
------	--------

Type

Comp

5.2 footage(name)

Description

Retrieves a footage item by name.

Parameters

name	String
------	--------

Type

Footage

5.3 thisProject

Description

Represents the project which contains the expression.

Type

Project

5.4 thisComp

Description

Represents the composition containing the expression.

Type

Comp

5.5 thisLayer

Description

Represents the layer containing the expression. Because thisLayer is the default object, its use is optional. For example, you can start an expression with thisLayer.width or width and get the same result.

Type

Layer, Light, or Camera

5.6 thisProperty

Description

Represents the property containing the expression. For example, if you write an expression on the Rotation property, you can start an expression with thisProperty to refer to the Rotation property.

Type

Property

5.7 time

Description

Represents the composition time, in seconds, at which the expression is being evaluated.

Type

Number

5.8 colorDepth

Description

Type the project color depth value. For example, colorDepth returns 16 when the project color depth is 16 bits per channel.

Type

Number

5.9 posterizeTime(updatesPerSecond)

Description

This expression allows you to set the frame rate for a property to be lower than the frame rate of the composition. The updatesPerSecond value passed in is *the number of times per second* the expression should evaluate.

Parameters

updatesPerSecond	Number
------------------	--------

Type

Number

Examples

To change a property to a random value 1 time per second:

```
posterizeTime(1);  
random()
```

To change a 2d property (such as Position or Scale) to a random value 3 times per second:

```
posterizeTime(3);  
var newValue = random(0, 100);  
[newValue, newValue];
```

To change a property to a random value within a specified range, every 12 frames:

```
var holdFrames = 12;
var minValue = 50;
var maxValue = 100;

posterizeTime(1 / framesToTime(holdFrames));

var newValue = random(minValue, maxValue);
newValue;
```

5.10 value

Description

Represents the value at the current time for the property containing the expression.

Type

Number, Array, or String

TIME CONVERSION

Note: If you want more control over the look of timecode in your footage, use the `timeToCurrentFormat` method or other `timeTo` methods to generate the timecode instead of using the `Timecode` or `Numbers` effect.

Create a text layer, add an expression to the Source Text property, and enter `timeToCurrentFormat()` in the expression field. With this method, you can format and animate the timecode text. In addition, the timecode uses the same display style defined by the current project settings.

6.1 `timeToFrames(t=time + thisComp.displayStartTime, fps=1.0 / thisComp.frameDuration, isDuration=false)`

Description

Converts the value of *t*, which defaults to the current composition time, to an integer number of frames. The number of frames per second is specified in the `fps` argument, which defaults to the frame rate of the current composition (`1.0 / thisComp.frameDuration`).

The `isDuration` argument, which defaults to `false`, should be `true` if the *t* value represents a difference between two times instead of an absolute time. Absolute times are rounded down toward negative infinity; durations are rounded away from zero (up for positive values).

Parameters

<code>t</code>	Number
<code>fps</code>	Number
<code>isDuration</code>	Boolean

Type

Number

6.2 framesToTime(frames, fps=1.0 / thisComp.frameDuration)

Description

The inverse of timeToFrames. Returns the time corresponding to the frames argument, which is required. It doesn't have to be an integer. See timeToFrames for explanation of the fps argument.

Parameters

frames	Number
fps	Number

Type

Number

6.3 timeToTimecode(t=time + thisComp.displayStartTime, timecodeBase=30, isDuration=false)

Description

Converts the value of t to a String representing timecode. See timeToFrames for an explanation of the t and isDuration arguments. The timecodeBase value, which defaults to 30, specifies the number of frames in one second.

Parameters

t	Number
timecodeBase	Number
isDuration	Boolean

Type

String

6.4 timeToNTSCTimecode(t=time + thisComp.displayStartTime, ntscDropFrame=false, isDuration=false)

Description

Converts t to a String representing NTSC timecode. See timeToFrames for an explanation of the t and isDuration arguments. If ntscDropFrame is false (the default), the result String is NTSC non-drop-frame timecode. If ntscDropFrame is true, the result String is NTSC drop-frame timecode.

Parameters

t	Number
ntscDropFrame	Boolean
isDuration	Boolean

Type

String

6.5 `timeToFeetAndFrames(t=time + thisComp.displayStartTime, fps=1.0 / thisComp.frameDuration, framesPerFoot=16, isDuration=false)`

Description

Converts the value of `t` to a `String` representing feet of film and frames. See `timeToFrames` for an explanation of the `t`, `fps`, and `isDuration` arguments. The `framesPerFoot` argument specifies the number of frames in one foot of film. It defaults to 16, which is the most common rate for 35mm footage.

Parameters

<code>t</code>	Number
<code>framesPerFoot</code>	Number
<code>isDuration</code>	Boolean

Type

String

6.6 `timeToCurrentFormat(t=time + thisComp.displayStartTime, fps=1.0 / thisComp.frameDuration, isDuration=false)`

Description

Converts the value of `t` to a `String` representing time in the current Project Settings display format. See `timeToFrames` for a definition of all the arguments.

Note: An optional `ntscDropFrame` argument was added to the `timeToCurrentFormat()` function in After Effects CS5.5 and later. Default: `ntscDropFrame=thisComp.ntscDropFrame`.

Parameters

<code>t</code>	Number
<code>fps</code>	Number
<code>isDuration</code>	Boolean

Type

String

VECTOR MATH

Vector Math functions are global methods that perform operations on arrays, treating them as mathematical vectors. Unlike built-in JavaScript methods, such as `Math.sin`, these methods are not used with the `Math` prefix. Unless otherwise specified, Vector Math methods are lenient about dimensions and return a value that is the dimension of the largest input Array object, filling in missing elements with zeros.

For example, the expression `add([10, 20], [1, 2, 3])` returns `[11, 22, 3]`.

Note: [JJ Gifford's website](#) provides explanations and examples that show how to use simple geometry and trigonometry with expressions.

7.1 `add(vec1, vec2)`

Description

Adds two vectors.

Parameters

<code>vec1</code>	Array
<code>vec2</code>	Array

Type

Array

7.2 `sub(vec1, vec2)`

Description

Subtracts two vectors.

Parameters

<code>vec1</code>	Array
<code>vec2</code>	Array

Type

Array

7.3 mul(vec, amount)

Description

Multiplies every element of the vector by the amount.

Parameters

vec	Array
amount	Number

Type

Array

7.4 div(vec, amount)

Description

Divides every element of the vector by the amount.

Parameters

vec	Array
amount	Number

Type

Array

7.5 clamp(value, limit1, limit2)

Description

The value of each component of `value` is constrained to fall between the values of the corresponding values of `limit1` and `limit2`.

Parameters

value	Number or Array
limit1	Number or Array
limit2	Number or Array

TypeNumber or Array

7.6 dot(vec1, vec2)

Description

Returns the dot (inner) product of the vector arguments.

Parameters

vec1	Array
vec2	Array

TypeNumber

7.7 cross(vec1, vec2)

Description

Returns the vector cross product of vec1 and vec2. Refer to a math reference or JavaScript guide for more information.

Parameters

vec1	Array (2- or 3-dimensional)
vec2	Array (2- or 3-dimensional)

TypeArray (2- or 3-dimensional)

7.8 normalize(vec)

Description

Normalizes the vector so that its length is 1.0. Using the normalize method is a short way of performing the operation `div(vec, length(vec))`.

Parameters

vec	Array
-----	-------

TypeArray

7.9 length(vec)

Description

Returns the length of vector `vec`.

Parameters

<code>vec</code>	Array
------------------	-------

Type

Number

7.10 length(point1, point2)

Description

Returns the distance between two points. The `point2` argument is optional.

For example, `length(point1, point2)` is the same as `length(sub(point1, point2))`.

For example, add this expression to the Focus Distance property of a camera to lock the focal plane to the camera's point of interest so that the point of interest is in focus:

```
length(position, pointOfInterest)
```

Parameters

<code>point1</code>	Array
<code>point2</code>	Array

Type

Number

7.11 lookAt(fromPoint, atPoint)

Description

The argument `fromPoint` is the location in world space of the layer you want to orient. The argument `atPoint` is the point in world space you want to point the layer at. The return value can be used as an expression for the Orientation property, making the z-axis of the layer point at `atPoint`.

This method is especially useful for cameras and lights. If you use this expression on a camera, turn off auto-orientation.

For example, this expression on the Orientation property of a spot light makes the light point at the anchor point of layer number 1 in the same composition:

```
lookAt(position, thisComp.layer(1).position)
```


Parameters

<code>fromPoint</code>	Array (3-dimensional)
<code>atPoint</code>	Array (3-dimensional)

Type

Array (3-dimensional)

RANDOM NUMBERS

Note: The wiggle method—which is used to randomly vary a property value—is in the Property attributes and methods category. See Property attributes and methods.

8.1 seedRandom(offset, timeless=false)

Description

The random and gaussRandom methods use a seed value that controls the sequence of numbers. By default, the seed is computed as a function of a unique layer identifier, the property within the layer, the current time, and an offset value of 0. Call seedRandom to set the offset to something other than 0 to create a different random sequence. Use true for the timeless argument to not use the current time as input to the random seed. Using true for the timeless argument allows you to generate a random number that doesn't vary depending on the time of evaluation. The offset value, but not the timeless value, is also used to control the initial value of the wiggle function.

For example, this expression on the Opacity property sets the Opacity value to a random value that does not vary with time:

```
seedRandom(123456, true);  
random()*100
```

The multiplication by 100 in this example converts the value in the range 0-1 returned by the random method into a number in the range 0-100; this range is more typically useful for the Opacity property, which has values from 0% to 100%.

Parameters

offset	Number
timeless	Boolean

Type

None

8.2 random()

Description

Returns a random number in the range 0-1.

Note: In After Effects CC and CS6, the behavior of random() is changed to be more random when layer IDs are close together. The wiggle() expression is not affected.

Type

Number

8.3 random(maxValOrArray)

Description

If maxValOrArray is a Number, this method returns a number in the range from 0 to maxValOrArray. If maxValOrArray is an Array, this method returns an Array with the same dimension as maxValOrArray, with each component ranging from 0 to the corresponding component of maxValOrArray.

Parameters

maxValOrArray	Number or Array
---------------	-----------------

Type

Number or Array

8.4 random(minValOrArray, maxValOrArray)

Description

If minValOrArray and maxValOrArray are Numbers, this method returns a number in the range from minValOrArray to maxValOrArray.

If the arguments are Arrays, this method returns an Array with the same dimension as the argument with the greater dimension, with each component in the range from the corresponding component of minValOrArray to the corresponding component of maxValOrArray.

For example, the expression random([100, 200], [300, 400]) returns an Array whose first value is in the range 100-300 and whose second value is in the range 200-400. If the dimensions of the two input Arrays don't match, higher-dimension values of the shorter Array are filled out with zeros.

Parameters

minValOrArray	Number or Array
maxValOrArray	Number or Array

Type

Number or Array

8.5 gaussRandom()

Description

The results have a Gaussian (bell-shaped) distribution. Approximately 90% of the results are in the range 0-1, and the remaining 10% are outside this range.

Type

Number

8.6 gaussRandom(maxValOrArray)

Description

When `maxValOrArray` is a `Number`, this method returns a random number. Approximately 90% of the results are in the 0 to `maxValOrArray` range, and the remaining 10% are outside this range.

When `maxValOrArray` is an `Array`, this method returns an `Array` of random values, with the same dimension as `maxValOrArray`. 90% of the values are in the range from 0 to `maxValOrArray`, and the remaining 10% are outside this range.

The results have a Gaussian (bell-shaped) distribution.

Parameters

<code>maxValOrArray</code>	Number or Array
----------------------------	-----------------

Type

Number or Array

8.7 gaussRandom(minValOrArray, maxValOrArray)

Description

If `minValOrArray` and `maxValOrArray` are `Numbers`, this method returns a random number. Approximately 90% of the results are in the range from `minValOrArray` to `maxValOrArray`, and the remaining 10% are outside this range.

If the arguments are `Arrays`, this method returns an `Array` of random numbers with the same dimension as the argument with the greater dimension. For each component, approximately 90% of the results are in the range from the corresponding component of `minValOrArray` to the corresponding component of `maxValOrArray`, and the remaining 10% are outside this range.

The results have a Gaussian (bell-shaped) distribution.

Parameters

minValOrArray	Number or Array
maxValOrArray	Number or Array

Type

Number or Array

8.8 noise(valOrArray)

Description

Returns a number in the range from -1 to 1. The noise is not actually random; it is based on Perlin noise, which means that the return values for two input values that are near one another tend to be near one another. This type of noise is useful when you want a sequence of seemingly random numbers that don't vary wildly from one to the other—as is usually the case when animating any apparently random natural motion.

Example:

```
rotation + 360*noise(time)
```

Parameters

valOrArray	Number or an Array (2- or 3-dimensional)
------------	--

Type

Number

INTERPOLATION

Description

For all the Interpolation methods, the argument `t` is often `time` or `value`, though it can have other values, instead. If `t` is `time`, the interpolation between values happens over a duration. If `t` is `value`, then the expression maps one range of values to a new range of values.

For additional explanations and examples of the Interpolation methods, see [JJ Gifford's website](#).

Chris and Trish Meyer provide additional information and examples for these methods in an article on the [ProVideo Coalition website](#).

Ian Haigh provides a script on [After Effects Scripts website](#) that you can use to easily apply advanced interpolation method expressions—such as bounces—to properties.

Andrew Devis provides a [pair of video tutorials](#) on the Creative COW website that show in detail how to use the linear expression method along with the Convert Audio To Keyframes command.

9.1 `linear(t, tMin, tMax, value1, value2)`

Description

Returns `value1` when `t <= tMin`. Returns `value2` when `t >= tMax`. Returns a linear interpolation between `value1` and `value2` when `tMin < t < tMax`.

For example, this expression on the Opacity property causes Opacity values to ramp linearly from 20% to 80% over the time from 0 seconds to 6 seconds:

```
linear(time, 0, 6, 20, 80)
```

This method—like all the Interpolation methods—can also be used to convert from one range of values to another.

For example, this expression on the Opacity property converts the Opacity values from the range 0%-100% to the range 20%-80%:

```
linear(value, 0, 100, 20, 80)
```

Parameters

<code>t</code>	Number
<code>tMin</code>	Number
<code>tMax</code>	Number
<code>value1</code>	Number or Array
<code>value2</code>	Number or Array

Type

Number or Array

9.2 linear(t, value1, value2)

Description

Returns a value that linearly interpolates from value1 to value2 as t ranges from 0 to 1. Returns value1 when t <= 0. Returns value2 when t >= 1.

Parameters

t	Number
value1	Number or Array
value2	Number or Array

Type

Number or Array

9.3 ease(t, tMin, tMax, value1, value2)

Description

Similar to linear with the same arguments, except that the interpolation eases in and out so that the velocity is 0 at the start and end points. This method results in a smooth animation.

Parameters

t	Number
tMin	Number
tMax	Number
value1	Number or Array
value2	Number or Array

Type

Number or Array

9.4 ease(t, value1, value2)

Description

Similar to linear with the same arguments, except that the interpolation eases in and out so that the velocity is 0 at the start and end points. This method results in a smooth animation.

Parameters

t	Number
value1	Number or Array
value2	Number or Array

Type

Number or Array

9.5 easeln(t, tMin, tMax, value1, value2)

Description

Similar to ease, except that the tangent is 0 only on the tMin side and interpolation is linear on the tMax side.

Parameters

t	Number
tMin	Number
tMax	Number
value1	Number or Array
value2	Number or Array

Type

Number or Array

9.6 easeln(t, value1, value2)

Description

Similar to ease, except that the tangent is 0 only on the value1 side and interpolation is linear on the value2 side.

Parameters

t	Number
value1	Number or Array
value2	Number or Array

Type

Number or Array

9.7 easeOut(t, tMin, tMax, value1, value2)

Description

Similar to ease, except that the tangent is 0 only on the tMax side and interpolation is linear on the tMin side.

Parameters

t	Number
tMin	Number
tMax	Number
value1	Number or Array
value2	Number or Array

Type

Number or Array

9.8 easeOut(t, value1, value2)

Description

Similar to ease, except that the tangent is 0 only on the value2 side and interpolation is linear on the value1 side.

Parameters

t	Number
value1	Number or Array
value2	Number or Array

Type

Number or Array

COLOR CONVERSION

Harry Frank provides a video tutorial on his [graymachine website](#) that shows how to use these color conversion methods to change the color of the waves produced by the Radio Waves effect.

10.1 rgbToHsl(rgbaArray)

Description

Converts a color in RGBA space to HSLA space. The input is an Array of normalized red, green, blue, and alpha channel values, all in the range of 0.0 to 1.0. The resulting value is an Array of hue, saturation, lightness, and alpha channel values, also in the range of 0.0 to 1.0.

Example:

```
rgbToHsl.effect("Change Color")("Color To Change")
```

Parameters

rgbaArray	Array (4-dimensional)
-----------	-----------------------

Type

Array (4-dimensional)

10.2 hslToRgb(hslaArray)

Description

Converts a color in HSLA space to RGBA space. This conversion is the opposite of the conversion performed by the rgbToHsl method.

Parameters

hslaArray	Array (4-dimensional)
-----------	-----------------------

Type

Array (4-dimensional)

10.3 hexToRgb(hexString)

Description

Converts a color in hex triplet space to RGB space, or in hex quartet space to RGBA space. For hex triplets, the alpha channel defaults to 1.0.

Available in After Effects 16.0 or later.

Parameters

hexString	String representing an hex triplet (6 digits, no alpha channel) or quartet (8 digits, includes alpha channel) containing only numerals or characters A–F. Optional leading characters 0x, 0X, or # will be ignored. Characters beyond 8 digits will be ignored.
-----------	---

Type

Array (4-dimensional)

Examples

Any of the following will return [1.0, 0.0, 1.0, 1.0]:

- `hexToRgb("FF00FF")`
- `hexToRgb("#FF00FF")`
- `hexToRgb("0xFF00FF")`
- `hexToRgb("0XFF00FFFF")` Note: This inputs an 8-digit hex quartet; the last two digits set alpha channel to 1.0.

11.1 degreesToRadians(degrees)

Description

Converts degrees to radians.

Parameters

degrees	Number
---------	--------

Type

Number

11.2 radiansToDegrees(radians)

Description

Converts radians to degrees.

Parameters

radians	Number
---------	--------

Type

Number

12.1 Project.fullPath

Description

The platform-specific absolute file path, including the project file name. If the project has not been saved, it returns an empty string.

Example:

```
thisProject.fullPath
```

Type

String

12.2 Project.bitsPerChannel

Description

The color depth of the project in bits per channel (bpc), as set in *Project Settings > Color Management*. They are one of 8, 16, or 32. Equivalent to the scripting project attribute *app.project.bitsPerChannel*.

Example:

```
thisProject.bitsPerChannel
```

Type

Number

12.3 Project.linearBlending

Description

The state of the Blend Colors Using 1.0 Gamma option in *Project Settings > Color Management*. Equivalent to the scripting project attribute *app.project.linearBlending*.

Example:

```
thisProject.linearBlending
```

Type

Boolean

13.1 `Comp.layer(index)`

Description

Retrieves the layer by number (order in the Timeline panel).

Example:

```
thisComp.layer(3)
```

Parameters

index	Number
-------	--------

Type

Layer, Light, or Camera

13.2 `Comp.layer(name)`

Description

Retrieves the layer by name. Names are matched according to layer name, or source name if there is no layer name. If duplicate names exist, After Effects uses the first (topmost) one in the Timeline panel.

Example:

```
thisComp.layer("Solid 1")
```

Parameters

name	String
------	--------

Type

Layer, Light, or Camera

13.3 `Comp.layer(otherLayer, relIndex)`

Description

Retrieves the layer that is `relIndex` layers above or below `otherLayer`. For example, `thisComp.layer(thisLayer, 1).active` returns true if the next layer down in the Timeline panel is active.

Parameters

<code>otherLayer</code>	Layer Object
<code>relIndex</code>	Number

Type

Layer, Light, or Camera

13.4 `Comp.layerByComment(comment)`

Description

Retrieves a layer by matching the `comment` parameter to the value in the layer's `Comment` column. The matches are simple text matches. They will match partial words, and are case sensitive. Matching does not appear to use regular expressions or wildcards. If duplicate comments exist, After Effects uses the first (topmost) one in the Timeline panel.

Example:

```
thisComp.layerByComment("Control") //note this will match a layer with a comment  
↳ "Controller" or "Motion Control"
```

Parameters

<code>comment</code>	String
----------------------	--------

Type

Layer, Light, or Camera

13.5 `Comp.marker`

Description

Note: You cannot access a composition marker by marker number. If you have a project created in a previous version of After Effects that uses composition marker numbers in expressions, you must change those calls to use `marker.key(name)` instead. Because the default name of a composition marker is a number, converting the reference to use the name is often just a matter of surrounding the number with quotation marks.

Type

MarkerProperty

13.6 Comp.marker.key(index)

Description

Returns the MarkerKey object of the marker with the specified index. The index refers to the order of the marker in composition time, not to the name of the marker.

For example, this expression returns the time of the first composition marker:

```
thisComp.marker.key(1).time
```

Parameters

index	Number
-------	--------

Type

MarkerKey

13.7 Comp.marker.key(name)

Description

Returns the MarkerKey object of the marker with the specified name. The name value is the name of the marker, as typed in the comment field in the marker dialog box, for example, marker.key("1"). For a composition marker, the default name is a number. If more than one marker in the composition has the same name, this method returns the marker that occurs first in time (in composition time). The value for a marker key is a String, not a Number.

For example, this expression returns the time of the composition marker with the name "0":

```
thisComp.marker.key("0").time
```

Parameters

name	String
------	--------

Type

MarkerKey

13.8 `Comp.marker.nearestKey(t)`

Description

Returns the marker that is nearest in time to `t`.

For example, this expression returns the time of the composition marker nearest to the time of 1 second:

```
thisComp.marker.nearestKey(1).time
```

This expression returns the time of the composition marker nearest to the current time:

```
thisComp.marker.nearestKey(time).time
```

Parameters

<code>t</code>	Number
----------------	--------

Type

MarkerKey

13.9 `Comp.marker.numKeys`

Description

Returns the total number of composition markers in the composition.

Type

Number

13.10 `Comp.numLayers`

Description

Returns the number of layers in the composition.

Type

Number

13.11 Comp.activeCamera

Description

Returns the Camera object for the camera through which the composition is rendered at the current frame. This camera is not necessarily the camera through which you are looking in the Composition panel.

Type

Camera

13.12 Comp.width

Description

Returns the composition width, in pixels. Apply the following expression to the Position property of a layer to center the layer in the composition frame: `[thisComp.width/2, thisComp.height/2]`

Type

Number

13.13 Comp.height

Description

Returns the composition height, in pixels.

Type

Number

13.14 Comp.duration

Description

Returns the composition duration, in seconds.

Type

Number

13.15 `Comp.ntscDropFrame`

Description

Returns true if the timecode is in drop-frame format.

Note: Available in After Effects CS5.5 and later.

Type

Boolean

13.16 `Comp.displayStartTime`

Description

Returns the composition start time, in seconds.

Type

Number

13.17 `Comp.frameDuration`

Description

Returns the duration of a frame, in seconds.

Type

Number

13.18 `Comp.shutterAngle`

Description

Returns the shutter-angle value of the composition, in degrees.

Type

Number

13.19 Comp.shutterPhase

Description

Returns the shutter phase of the composition, in degrees.

Type

Number

13.20 Comp.bgColor

Description

Returns the background color of the composition.

Type

Array (4-dimensional)

13.21 Comp.pixelAspect

Description

Returns the pixel aspect ratio of the composition.

Type

Number

13.22 Comp.name

Description

Returns the name of the composition.

Type

String

Description

To use a footage item from the Project panel as an object in an expression, use the global footage method, as in `footage("file_name")`. You can also access a footage object using the source attribute on a layer whose source is a footage item.

14.1 Footage.width

Description

Returns the width of the footage item, in pixels.

Type

Number

14.2 Footage.height

Description

Returns the height of the footage item, in pixels.

Type

Number

14.3 Footage.duration

Description

Returns the duration of the footage item, in seconds.

Type

Number

14.4 Footage.frameDuration

Description

Returns the duration of a frame in the footage item, in seconds.

Type

Number

14.5 Footage.ntscDropFrame

Description

Returns true if the timecode is in drop-frame format. (After Effects CS5.5 and later.)

Type

Boolean

14.6 Footage.pixelAspect

Description

Returns the pixel aspect ratio of the footage item.

Type

Number

14.7 Footage.name

Description

Returns the name of the footage item as shown in the Project panel.

Type

String

14.8 Footage.sourceText

Description

Returns the contents of a JSON file as a string.

The `eval()` method can be used to convert the string to an array of `sourceData` objects, identical to the results of the `Footage.sourceData` attribute, from which the individual data streams can be referenced as hierarchal attributes of the data.

For example:

```
var myData = eval(footage("sample.json").sourceText);
myData.sampleValue;
```

Type

String, the contents of the JSON file; read-only.

14.9 Footage.sourceData

Description

Returns the data of a JSON file as an array of `sourceData` objects.

The structure of the JSON file will determine the size and complexity of the array.

Individual data streams can be referenced as hierarchal attributes of the data.

For example, given a data stream named “Color”, the following will return the value of “Color” from the first data object:

```
footage("sample.json").sourceData[0].Color
```

Typical use is to assign a JSON file’s `sourceData` to a variable, and then reference the desired data stream. For example:

```
var myData = footage("sample.json").sourceData;
myData[0].Color;
```

Type

An array of `sourceData` objects; read-only.

14.10 Footage.dataValue(dataPath)

Description

Returns the value of specified static or dynamic data stream in a MGJSON file.

Accepts a single array value to define the path in the hierarchy to the desired data stream.

For example, to return data of the first child:

```
footage("sample.mgjson").dataValue([0])
```

Or to return data of the first child in the second group:

```
footage("sample.mgjson").dataValue([1][0])
```

Parameters

dataPath	Array, required. The path in the hierarchy to a static or dynamic data stream.
----------	--

Type

The value of the data stream.

14.11 Footage.dataKeyCount(dataPath)

Description

Returns the number of samples in a specified dynamic data stream in a MGJSON file.

Accepts a single array value to define the path in the hierarchy to the desired dynamic data stream.

For example, to return the count of samples for the first child:

```
footage("sample.mgjson").dataKeyCount([0])
```

Or to return the count of samples for the second group:

```
footage("sample.mgjson").dataKeyCount([1][0])
```

Parameters

dataPath	Array, required. The path in the hierarchy to a static or dynamic data stream.
----------	--

Type

The number of samples in the dynamic data stream.

14.12 Footage.dataKeyTimes(dataPath, t0 = startTime, t1 = endTime)

Description

Returns the time in seconds for the samples of a specified dynamic data stream in a MGJSON file.

Optionally specify the time span from which to return samples. By default the time for all samples between `startTime` and `endTime` in the dynamicdata stream are returned, as defined by the data stream's `samplesTemporalExtent` property in the MGJSON file.

Accepts a single array value to define the path in the hierarchy to the desired dynamic data stream.

The following example returns the times of samples between 1 second and 3 seconds for the first child:

```
footage("sample.mgjson").dataKeyTimes([0], 1, 3)
```

Parameters

<code>dataPath</code>	Array, required. The path in the hierarchy to a static or dynamic data stream.
<code>t0</code>	Number, optional. The start time, in seconds, of the span from which to return samples. Defaults to <code>startTime</code> .
<code>t1</code>	Number, optional. The end time, in seconds, of the span from which to return samples. Defaults to <code>endTime</code> .

Type

Array of numbers representing the sample times.

14.13 Footage.dataKeyValues(dataPath, t0 = startTime, t1 = endTime)

Description

Returns the values for the samples of a specified dynamic data stream in a MGJSON file.

Optionally specify the time span from which to return samples. By default the time for all samples between `startTime` and `endTime` in the dynamicdata stream are returned, as defined by the data stream's `samplesTemporalExtent` property in the MGJSON file.

Accepts a single array value to define the path in the hierarchy to the desired dynamic data stream.

The following example returns the values of samples between 1 second and 3 seconds for the first child:

```
footage("sample.mgjson").dataKeyValues([0], 1, 3)
```

Parameters

<code>dataPath</code>	Array, required. The path in the hierarchy to a static or dynamic data stream.
<code>t0</code>	Number, optional. The start time, in seconds, of the span from which to return samples. Defaults to <code>startTime</code> .
<code>t1</code>	Number, optional. The end time, in seconds, of the span from which to return samples. Defaults to <code>endTime</code> .

Type

Array of numbers representing the sample values.

CAMERA

Camera objects have the same attributes and methods as Layer objects, except for:

- source
- effect
- mask
- width
- height
- anchorPoint
- scale
- opacity
- audioLevels
- timeRemap
- all the material properties

15.1 Camera.pointOfInterest

Description

Returns the point of interest values of a camera in world space.

Type

Array (3 dimensional)

15.2 Camera.zoom

Description

Returns the zoom values of a camera in pixels.

Here's an expression for the Scale property of a layer that maintains the relative size of the layer in frame while changing the z position (depth) of a layer or the Zoom value of a camera:

```
cam = thisComp.activeCamera;  
distance = length(sub(position, cam.position));  
scale * distance / cam.zoom;
```

Type

Number

15.3 Camera.depthOfField

Description

Returns 1 if the Depth Of Field property of a camera is on, or returns 0 if the Depth Of Field property is off.

Type

Boolean Number

15.4 Camera.focusDistance

Description

Returns the focus distance value of a camera, in pixels.

Type

Number

15.5 Camera.aperture

Description

Returns the aperture value of a camera, in pixels.

Type

Number

15.6 Camera.blurLevel

Description

Returns the blur level value of a camera as a percentage.

Type

Number

15.7 Camera.irisShape

Description

Returns the iris shape value from 1-10, corresponding to the selected dropdown value.

Type

Number

15.8 Camera.irisRotation

Description

Returns the iris rotation value, in degrees.

Type

Number

15.9 Camera.irisRoundness

Description

Returns the camera iris roundness value as a percentage.

Type

Number

15.10 Camera.irisAspectRatio

Description

Returns the camera iris aspect ratio, from 1 to 100.

Type

Number

15.11 Camera.irisDiffractionFringe

Description

Returns the camera iris diffraction fringe, from 1 to 100.

Type

Number

15.12 Camera.highlightGain

Description

Returns the camera highlight gain, from 1 to 100.

Type

Number

15.13 Camera.highlightThreshold

Description

Returns the camera highlight threshold.

- In an 8-bit comp, this value ranges from 0 to 100
- In a 16-bit comp, this value ranges from 0 to 32768
- In a 32-bit comp, this value ranges from 0 to 1.0

Type

Number

15.14 Camera.highlightSaturation

Description

Returns the camera highlight saturation, from 1 to 100.

Type

Number

15.15 Camera.active

Description

Returns `true` if the camera:

1. is the active camera for the composition at the current time: the *video switch* for the camera layer is on
2. the current time is in the range from the *in point* of the camera layer to the *out point* of the camera layer
3. and it is the first (topmost) such camera layer listed in the *timeline panel*

Returns `false` otherwise.

Type

Boolean

LIGHT

Description

Light objects have the same attributes and methods as Layer objects, except for:

- source
- effect
- mask
- width
- height
- anchorPoint
- scale
- opacity
- audioLevels
- timeRemap
- all the material properties

Note: David Van Brink provides an instructional article and sample project on his [omino pixel blog](#) that shows how to use expressions with lights.

16.1 Light.pointOfInterest

Description

Returns the point of interest values for a light in world space.

Type

Array (3-dimensional)

16.2 Light.intensity

Description

Returns the intensity values of a light as a percentage.

Type

Number

16.3 Light.color

Description

Returns the color value of a light.

Type

Array (4-dimensional)

16.4 Light.coneAngle

Description

Returns the cone angle of a light, in degrees.

Type

Number

16.5 Light.coneFeather

Description

Returns the cone feather value of a light as a percentage.

Type

Number

16.6 Light.shadowDarkness

Description

Returns the shadow darkness value of a light as a percentage.

Type

Number

16.7 Light.shadowDiffusion

Description

Returns the shadow diffusion value of a light, in pixels.

Type

Number

GROUP

All properties in the Timeline are organized into groups, which share some attributes of properties like name and `propertyIndex`. Groups can have a fixed number of properties (e.g. an individual effect whose properties don't change) or a variable number of properties (e.g. the Effects group itself which can have any number of effect within it).

- **Top-level groups in a Layer:**

- Motion Trackers
- Text
- Contents
- Masks
- Effects
- Transform
- Layer Styles
- Geometry Options
- Material Options
- Audio
- Data
- Essential Properties

- **Nested groups**

- Individual effects
 - Individual masks
 - Shape groups
 - Text Animators
-

17.1 numProperties

Description

Returns the number of properties or groups directly within a group. This does not include properties nested inside child groups.

Tip: Find the number of effects applied to a layer with `thisLayer("ADBE Effect Parade").numProperties` using the match name to remain language-agnostic.

Type

Number

17.2 propertyGroup(countUp=1)

Description

Returns a higher-level property group relative to the property group on which the method is called. See *propertyGroup(countUp=1)* for additional details.

Type

Group

17.3 propertyIndex

Description

Returns the index of a property group relative to other properties or groups in its property group.

Type

Number

17.4 name

Description

Returns the name of the property group.

Type

String

18.1 Effect.active

Description

Returns `true` if the effect is turned on (the *effect switch* is selected).

Type

Boolean

18.2 Effect.param(name)

Description

Returns a property within an effect. Effect control points are always in layer space.

Example:

```
effect("Bulge").param("Bulge Height")
```

Parameters

name	String
------	--------

Type

Property

18.3 Effect.param(index)

Description

Returns a property within an effect. Effect control points are always in layer space. For example, `effect("Bulge").param(4)` returns the Bulge Height property.

Parameters

index	Number
-------	--------

Type

Property

Note: You can link Mask Path properties to other path properties (paths in a shape layer and brush strokes), but the properties are not accessible for direct numerical manipulation through expressions.

19.1 Mask.maskOpacity

Description

Returns the opacity value of a mask as a percentage.

Type

Number

19.2 Mask.maskFeather

Description

Returns the feather value of a mask, in pixels.

Type

Number

19.3 Mask.maskExpansion

Description

Returns the expansion value of a mask, in pixels.

Type

Number

19.4 Mask.invert

Description

Returns `true` if the mask is inverted or `false` if it is not.

Type

Boolean

PROPERTY

Example: Animating with the `propertyGroup` method and `propertyIndex` attribute

[image]

In this example, the `propertyGroup` method for each brush stroke targets the Brush property group because that group is two property groups up from the Rotation property. The `propertyIndex` attribute in each Brush stroke then returns a unique value for each Brush stroke. The resulting value is then multiplied by the `time` and `200` and applied to each rotation value, rotating each brush stroke differently, creating swirling paint strokes:

```
propertyGroup(2).propertyIndex * time * 200  
propertyGroup(2).propertyIndex * time * 200
```

[image]

20.1 value

Description

Returns the value of a property at the current time.

Type

Number, Array, or String

20.2 valueAtTime(*t*)

Description

Returns the value of a property at the specified time, in seconds.

For example, to have a property value for each frame chosen randomly from a set of four values, set your four values as keyframes at 0, 1, 2, and 3 seconds, and then apply the following expression to the property:

```
valueAtTime(random(4))
```

Note: Dan Ebberts provides more examples and techniques for using the `valueAtTime` and `velocityAtTime` methods on [MotionScript](#).

Parameters

t	Number
---	--------

Type

Number or Array

20.3 velocity

Description

Returns the temporal velocity value at the current time. For spatial properties, such as Position, it returns the tangent vector value. The result is the same dimension as the property.

Type

Number or Array

20.4 velocityAtTime(t)

Description

Returns the temporal velocity value at the specified time.

Parameters

t	Number
---	--------

Type

Number or Array

20.5 speed

Description

Returns a 1D, positive speed value equal to the speed at which the property is changing at the default time. This element can be used only for spatial properties.

Type

Number

20.6 speedAtTime(t)

Description

Returns the spatial speed value at the specified time.

Parameters

t	Number
---	--------

Type

Number

20.7 wiggle(freq, amp, octaves=1, amp_mult=0.5, t=time)

Description

Randomly shakes (wiggles) the value of the property.

freq value is the frequency in wiggles per second.

amp value is the amplitude in units of the property to which it is applied.

octaves is the number of octaves of noise to add together. This value controls how much detail is in the wiggle. Make this value higher than the default of 1 to include higher frequencies or lower to include amplitude harmonics in the wiggle.

amp_mult is the amount that amp is multiplied by for each octave. This value controls how fast the harmonics drop off. The default is 0.5; make it closer to 1 to have the harmonics added at the same amplitude as the base frequency, or closer to 0 to add in less detail.

t is the base start time. This value defaults to the current time. Use this parameter if you want the output to be a wiggle of the property value sampled at a different time.

Example:

```
position.wiggle(5, 20, 3, 0.5)
```

This produces about 5 wiggles per second with an average size of about 20 pixels. In addition to the main wiggle, two more levels of detailed wiggles occur with a frequency of 10 and 20 wiggles per second, and sizes of 10 and 5 pixels, respectively.

This example, on a two-dimensional property such as Scale, wiggles both dimensions by the same amount:

```
v = wiggle(5, 10);
[v[0], v[0]]
```

This example, on a two-dimensional property, wiggles only along the y-axis:

```
freq = 3;
amp = 50;
w = wiggle(freq, amp);
[value[0], w[1]];
```

Note: Dan Ebberts provides an example expression and a detailed explanation on his [website](#) that shows how to use the time parameter of the wiggle method to create a looping animation.

Parameters

freq	Number
amp	Number
octaves	Number
amp_mult	Number
t	Number

Type

Number or Array

20.8 temporalWiggle(freq, amp, octaves=1, amp_mult=0.5, t=time)

Description

Samples the property at a wiggled time.

freq value is the frequency in wiggles per second.

amp is the amplitude in units of the property to which it is applied.

octaves is the number of octaves of noise to add together.

amp_mult is the amount that amp is multiplied by for each octave

t is the base start time.

For this function to be meaningful, the property it samples must be animated, because the function alters only the time of sampling, not the value.

Example:

```
scale.temporalWiggle(5, 0.2)
```

Parameters

freq	Number
amp	Number
octaves	Number
amp_mult	Number
t	Number

Type

Number or Array

20.9 smooth(width=.2, samples=5, t=time)

Description

Smooths the property values over time, converting large, brief deviations in the value to smaller, more evenly distributed deviations. This smoothing is accomplished by applying a box filter to the value of the property at the specified time. The width value is the range of time (in seconds) over which the filter is averaged. The samples value is the number of discrete samples evenly spaced over time; use a larger value for greater smoothness (but decreased performance). Generally, you'll want samples to be an odd number so that the value at the current time is included in the average.

Example:

```
position.smooth(0.1, 5)
```

Parameters

width	Number
samples	Number
t	Number

Type

Number or Array

20.10 loopIn(type="cycle", numKeyframes=0)

Description

Loops a segment of time that is measured from the first keyframe on the layer forward toward the Out point of the layer. The loop plays from the In point of the layer. The numKeyframes value determines what segment is looped: The segment looped is the portion of the layer from the first keyframe to the numKeyframes+1 keyframe. For example, loopIn("cycle", 3) loops the segment bounded by the first and fourth keyframes. The default value of 0 means that all keyframes loop. You can use keyframe-looping methods to repeat a series of keyframes. You can use these methods on most properties. Exceptions include properties that can't be expressed by simple numeric values in the Timeline panel, such as the Source Text property, path shape properties, and the Histogram property for the Levels effect. Keyframes or duration values that are too large are clipped to the maximum allowable value. Values that are too small result in a constant loop.

type	result
cycle	(default) Repeats the specified segment.
ping-pong	Repeats the specified segment, alternating between forward and backward.
offset	Repeats the specified segment, but offsets each cycle by the difference in the value of the property at the start and end of the segment, multiplied by the number of times the segment has looped.
continue	Does not repeat the specified segment, but continues to animate a property based on the velocity at the first or last keyframe. For example, if the last keyframe of a Scale property of a layer is 100%, the layer continues to scale from 100% to the Out point, instead of looping directly back to the Out point. This type does not accept a keyframes or duration argument.

Tip: Use `loopIn("continue") + loopOut("continue") - value` to have a continued motion before and after the property's keyframes. *Tip from Paul Slemmer.*

Type

Number or Array

20.11 `loopOut(type="cycle", numKeyframes=0)`

Description

Loops a segment of time that is measured from the last keyframe on the layer back toward the In point of the layer. The loop plays until the Out point of the layer. The specified number of keyframes determines the segment to loop. The `numKeyframes` value sets the number of keyframe segments to loop; the specified range is measured backward from the last keyframe.

For example, `loopOut("cycle", 1)` loops the segment bounded by the last keyframe and second-to-last keyframe. The default value of 0 means that all keyframes loop. See the entry for `loopIn` for more information.

Note: David Van Brink provides an instructional article and sample project on his [omino pixel blog](#) that show how to use the Echo effect, the Particle Playground effect, and the `loopOut` method to animate a swarm of stylized swimming bacteria.

Type

Number or Array

20.12 `loopInDuration(type="cycle", duration=0)`

Description

Loops a segment of time that is measured from the first keyframe on the layer forward toward the Out point of the layer. The loop plays from the In point of the layer. Specified duration determines the segment to loop. The duration value sets the number of composition seconds in a segment to loop; the specified range is measured from the first keyframe.

For example, `loopInDuration("cycle", 1)` loops the first second of the entire animation. The default of 0 means that the segment to loop begins at the layer Out point. See the entry for `loopIn` for more information.

Type

Number or Array

20.13 loopOutDuration(type="cycle", duration=0)

Description

Loops a segment of time that is measured from the last keyframe on the layer back toward the In point of the layer. The loop plays until the Out point of the layer. Specified duration determines the segment to loop. The duration value sets the number of composition seconds in a segment to loop; the specified range is measured backward from the last keyframe.

For example, `loopOutDuration("cycle", 1)` loops the last second of the entire animation. The default of `0` means that the segment to loop begins at the layer In point. See the entry for `loopIn` for more information.

Type

Number or Array

20.14 key(index)

Description

Returns the Key or MarkerKey object by number.

For example, `key(1)` returns the first keyframe.

Parameters

index	Number
-------	--------

Type

Key or MarkerKey

20.15 key(markerName)

Description

Returns the MarkerKey object with this name. Use only on marker properties.

markerName	String
------------	--------

Type

MarkerKey

20.16 nearestKey(τ)

Description

Returns the Key or MarkerKey object nearest to a designated time τ .

Parameters

τ	Number
--------	--------

Type

Key or MarkerKey

20.17 numKeys

Description

Returns the number of keyframes on a property. Returns the number of markers on a marker property.

Note: If you use the Separate Dimensions command to separate the dimensions of the Position property into individual components, the number of keyframes changes, so the value returned by this method changes.

Type

Number

20.18 propertyGroup(countUp=1)

Description

Returns a group of properties relative to the property on which the expression is written.

For example, if you add the `propertyGroup(1)` expression to the Rotation property of a brush stroke, the expression targets the Transform property group, which contains the Rotation property. If you add `propertyGroup(2)` instead, the expression targets the Brush property group.

This method lets you establish name-independent relationships in the property hierarchy. It is especially useful when duplicating properties that contain expressions. The `numProperties` method for `propertyGroup` returns the number of properties in the property group.

This example returns the number of properties in the group that contains the property on which the expression is written:

```
thisProperty.propertyGroup(1).numProperties
```

Type

Group

20.19 propertyIndex

Description

Returns the index of a property relative to other properties in its property group, including property groups within masks, effects, text animators, selectors, shapes, trackers, and track points.

Type

Number

20.20 name

Description

Returns the name of the property or property group.

Type

String

PATH PROPERTY

Note: Available in After Effects CC18 and later.

Example 1

Writes the list of point and tangent coordinates from *Path 1* of *Shape 1* on layer *Shape Layer 1*, at `time=0`, into a string. Apply this to the source text property of a text layer for a readout of the coordinates and incoming and outgoing tangents of the shape.

```
pointsList = "";
sampleTime = 0;
myShape = thisComp.layer("Shape Layer 1").content("Shape 1").content("Path 1").path;

for (i = 0; i < myShape.points(sampleTime).length; i++) {
    pointsList += "c: " + myShape.points(sampleTime)[i].toString() + " i: " +
    ↪myShape.inTangents(sampleTime)[i].toString() + " o: " + myShape.
    ↪outTangents(sampleTime)[i].toString() + "\n";
}

pointsList;
```

Example 2

Reads the coordinates of the first vertex of *Mask 1* on *Dark Gray Solid 1* and converts them to composition coordinates. Apply this to a 2D point control of an effect, such as *Write-on* or *CC Particle Systems II*, to make the effect trace or track the first point of an animated mask. Duplicate the effect and change the path points index value ([0]) to trace or track the other points of the mask.

```
myLayer = thisComp.layer("Dark Gray Solid 1");
myLayer.toComp(myLayer.mask("Mask 1").maskPath.points()[0]);
```

21.1 name

Description

Returns the name of the property.

Type

String

21.2 PathProperty.points(t=time)

Description

Get the x,y coordinates of all points on a path. Coordinates for layer mask path points are relative to the layer's origin in its upper-left hand corner. Coordinates for Bezier shape path points are relative to the anchor point of the path's shape group (ex., "Transform: Shape 1 > Anchor Point"). Coordinates for brush stroke path points are relative to the start of the stroke; the first point is [0,0]. This method can be passed into the createPath() method for the points parameter when duplicating a path.

Optionally specify the time at which sample to the path.

Parameters

t	Number
---	--------

Type

Array of number pair arrays, rounded to the fourth decimal place

21.3 PathProperty.inTangents(t=time)

Description

Get the x,y coordinates of the incoming tangent handle for all points on a path. Tangent coordinate values are offset relative to the parent point's coordinates. i.e., The value [0,0] creates no curvature at the incoming tangent. This method can be passed into the createPath() method for the inTangents parameter when duplicating a path.

Optionally specify the time at which sample to the path.

Type

Array of number pair arrays, rounded to the fourth decimal place

21.4 PathProperty.outTangents(t=time)

Description

Get the x, y coordinates of the outgoing tangent handle for all points on a path. Tangent coordinate values are offset relative to the parent point's coordinates. i.e., The value `[0,0]` creates no curvature at the outgoing tangent. This method can be passed into the `createPath()` method for the `outTangents` parameter when duplicating a path.

Optionally specify the time at which sample to the path.

Parameters

t	Number
---	--------

Type

Array of number pair arrays, rounded to the fourth decimal place

21.5 PathProperty.isClosed()

Description

Determines if the path is open or closed. Returns `true` if the path is closed, `false` if the path is open. This method can be passed into the `createPath()` method for the `is_closed` parameter when duplicating a path.

Type

Boolean

21.6 PathProperty.pointOnPath(percentage=0.5, t=time)

Description

Get the x, y coordinates of an arbitrary point along a path. The point is expressed as a percentage of the arc-length of the path. 0% is the first point and 100% is the last point. When the path is closed, 0% and 100% will return the same coordinates. Percentage of arc-length is used to ensure uniform speed along the path. Other than 0% and 100%, percentages do not necessarily correlate with the Bezier points on the path. (i.e., For a path with three points, the second point will not necessarily be at 50%.) This also means that for an open path and closed path with identical points, the percentage along the open path will not return the same coordinates as the closed path due to the additional length of the closed path.

Optionally specify the time at which sample to the path.

Parameters

percentage	Number
t	Number

Type

A number pair array

21.7 PathProperty.tangentOnPath(percentage=0.5, t=time)

Description

Get the calculated x,y coordinates of the outgoing tangent handle for an arbitrary point along a path. Tangent coordinate values are offset relative to the parent point's coordinates. i.e., The value [0,0] creates no curvature at the outgoing tangent. The incoming tangent handle is the inverse of this value (multiply the x,y coordinates by -1). The tangent's parent point is expressed as a percentage of the arc-length of the path. Read the description of the `pointOnPath()` method for details about arc-length percentage. The coordinates returned by `tangentOnPath()` are calculated from its parent point and will differ from those returned by `outTangents()` if a user-defined point also exists at that arc-length percentage. The linear distance between the parent point's coordinates and `tangentOnPath()` coordinates will always be 1. You can multiply the returned coordinates to create a longer tangent, for example `(myPath.tangentOnPath() * 100)`.

Optionally specify the time at which sample to the path.

Parameters

percentage	Number
t	Number

Type

A number pair array

21.8 PathProperty.normalOnPath(percentage=0.5, t=time)

Description

Get the calculated x,y coordinates of the normal for an arbitrary point along a path. Coordinate values of normals are offset relative to the parent point's coordinates. i.e., The value [0,0] is the same as the parent point. The normal's parent point is expressed as a percentage of the arc-length of the path. Read the description of the `pointOnPath()` method for details about arc-length percentage. The coordinates returned by `normalOnPath()` are calculated from its parent point. The linear distance between the parent point's coordinates and `normalOnPath()` coordinates will always be 1. You can multiply the returned coordinates to create a longer normal, for example. `(myPath.normalOnPath() * 100)`.

Optionally specify the time at which sample to the path.

Parameters

percentage	Number
t	Number

Type

A number pair array

21.9 PathProperty.createPath(points=[[0,0], [100,0], [100,100], [0,100]], inTangents=[], outTangents=[], is_closed=true)

Description

Creates a path object from a set of points and tangents. The points are defined by an array of number pair arrays representing their x, y coordinates. The array length must be at least 1, and can be of any greater length. The incoming and outgoing tangent handles of the points are defined by an array of number pair arrays representing their x, y offset coordinates. The length of the tangent arrays must be exactly the same as the points parameter. Tangent coordinate values are offset relative to the parent point's coordinates. i.e., The value [0,0] creates no curvature at the incoming tangent. The `points()`, `inTangents()`, `outTangents()`, and `isClosed()` methods of a path can be passed into the `points`, `inTangents`, `outTangents`, and `is_closed` parameters to duplicate a path. The points and tangents of the same path can be passed into `createPath()` with modifications to generate a different result.

For example, the following expression will remove curves from Mask 1 by not passing the `inTangents` or `outTangents` parameters:

```
myMask = mask("Mask 1").path;
myMask.createPath(myMask.points());
```

The following example passes the points and tangents of *Mask 1* and converts it to an open path by setting `is_closed` to false:

```
myMask = mask("Mask 1").path;
myMask.createPath(myMask.points(), myMask.inTangents(), myMask.outTangents(), false);
```

Parameters

<code>points</code>	Array
<code>inTangents</code>	Array
<code>outTangents</code>	Array
<code>is_closed</code>	Boolean

Type

Path

KEY

Description

When you access a Key object, you can get time, index, and value properties from it. For example, the following expression gives you the value of the third Position keyframe: `position.key(3).value`.

The following expression, when written on an Opacity property with keyframes, ignores the keyframe values and uses only the placement of the keyframes in time to determine where a flash should occur:

```
d = Math.abs(time - nearestKey(time).time);  
easeOut(d, 0, .1, 100, 0)
```

22.1 Key.value

Description

Returns the value of the keyframe.

Type

Number or Array

22.2 Key.time

Description

Returns the time of the keyframe.

Type

Number

22.3 Key.index

Description

Returns the index of the keyframe.

Type

Number

MARKERKEY

You can access values for composition markers and layer markers using the same methods. Access layer markers through the `thisLayer.marker` object; access composition markers through the `thisComp.marker` object.

For the purpose of expressions, markers are a special type of `Key` object, so you can use methods such as `nearestKey(time)` to access markers, and markers also have time and index attributes. The index attribute is not the number (name) of the marker; it is the keyframe index number, representing the order of the marker in the time ruler.

Expressions have access to all the values for a marker that you can set in the Composition Marker or Layer Marker dialog box.

This expression on the Source Text property of a text layer displays the time, duration, index, comment (name), chapter, URL, frame target, and cue point name for the layer marker nearest the current time, and whether the marker is for an event cue point:

```
m = thisLayer.marker.nearestKey(time);
s = "time:" + timeToCurrentFormat(m.time) + "\r" +
"duration: " + m.duration + "\r" +
"key index: " + m.index + "\r" +
"comment:" + m.comment + "\r" +
"chapter:" + m.chapter + "\r" +
"URL:" + m.url + "\r" +
"frame target: " + m.frameTarget + "\r" +
"cue point name: " + m.cuePointName + "\r" +
"Event cue point? " + m.eventCuePoint + "\r";
for (param in m.parameters){
s += "parameter: " + param + " value: " + m.parameters[param] + "\r";
}
s
```

Because the XMP metadata in a footage item can be converted into layer markers for a layer based on that item, expressions can interact with XMP metadata. For information, see [XMP metadata in After Effects](#).

Dan Ebberts provides a tutorial on the [After Effects Developer Center](#) that includes an example of using XMP metadata with expressions.

23.1 Marker.duration

Description

Duration, in seconds, of marker.

Type

Number

23.2 Marker.comment

Description

Contents of Comment field in marker dialog box.

Type

String

23.3 Marker.chapter

Description

Contents of Chapter field in marker dialog box.

Type

String

23.4 Marker.url

Description

Contents of URL field in marker dialog box.

Type

String

23.5 Marker.frameTarget

Description

Contents of Frame Target field in marker dialog box.

Type

String

23.6 Marker.eventCuePoint

Description

Setting for cue point type in marker dialog box. True for Event; false for Navigation.

Type

Boolean

23.7 Marker.cuePointName

Description

Contents of cue point Name field in marker dialog box.

Type

String

23.8 Marker.parameters

Description

Contents of Parameter Name and Parameter Value fields in marker dialog box.

For example, if you have a parameter named background color, then you can use the following expression to access its value at the nearest marker:

```
thisComp.marker.nearestKey(time).parameters["background color"]
```

Type

Associative array of String values

23.9 Marker.protectedRegion

Description

State of the Protected Region option in the Composition Marker dialog box. When true, the composition marker behaves as a protected region. Will also return true for protected region markers on nested composition layers, but is otherwise not applicable to layer markers.

Available in After Effects 16.0 or later.

Type

Boolean

LAYER SUB-OBJECTS

Note: For After Effects CC and CS6, the Expression language menu, the “Layer Sub-objects”, “Layer General”, “Layer Properties”, “Layer 3D”, and “Layer Space Transforms” have been arranged into a “Layer” submenu.

24.1 Layer.source

Description

Returns the source Comp or source Footage object for the layer. Default time is adjusted to the time in the source.

Example:

```
source.layer(1).position
```

Type

Comp or Footage

24.2 Layer.sourceTime(t=time)

Description

Returns the layer source corresponding to time *t*.

Note: After Effects CS5.5 and later

Parameters

t	Number
---	--------

Type

Number

24.3 Layer.sourceRectAtTime(*t* = time, includeExtents = false)

Description

Returns a JavaScript object with four attributes: [top, left, width, height]

Extents apply only to shape layers and paragraph text layers.

Shape layer extents increase the size of the layer bounds as necessary.

Paragraph text layers returns the bounds of the paragraph box.

Note: After Effects 13.2 and later. Paragraph text extents added in After Effects 15.1.

Example:

```
myTextLayer.sourceRectAtTime().width
```

Parameters

<i>t</i>	Number
<i>includeExtents</i>	Bool

Type

Array (4-dimensional)

24.4 Layer.effect(name)

Description

After Effects finds the effect by its name in the Effect Controls panel. The name can be the default name or a user-defined name. If multiple effects have the same name, the effect closest to the top of the Effect Controls panel is used.

Example:

```
effect("Fast Blur")("Blurriness")
```

Parameters

name	String
------	--------

Type

Effect

24.5 Layer.effect(index)

Description

After Effects finds the effect by its index in the Effect Controls panel, starting at 1 and counting from the top.

Parameters

index	Number
-------	--------

Type

Effect

24.6 Layer.mask(name)

Description

The name can be the default name or a user-defined name. If multiple masks have the same name, the first (topmost) mask is used.

Example:

```
mask("Mask 1")
```

Parameters

name	String
------	--------

Type

Mask

24.7 Layer.mask(index)

Description

After Effects finds the mask by its index in the Timeline panel, starting at 1 and counting from the top.

Parameters

index	Number
-------	--------

Type

Mask

LAYER GENERAL

25.1 Layer.width

Description

Returns the width of the layer, in pixels. It is the same as `source.width`.

Type

Number

25.2 Layer.height

Description

Returns the height of the layer, in pixels. It is the same as `source.height`.

Type

Number

25.3 Layer.index

Description

Returns the index number of the layer in the composition.

Type

Number

25.4 Layer.parent

Description

Returns the parent Layer object of the layer, if it has one.

Example:

```
position[0] + parent.width
```

Type

Layer, Light, or Camera

25.5 Layer.hasParent

Description

Returns true if the layer has a parent or false if it doesn't. Use the hasParent attribute to determine if a layer has a parent layer. You can use this attribute even if the layer has no parent layer at present. For example, the following expression indicates that the layer to which you apply it wiggles based on the position of the parent. If the layer has no parent, then it wiggles based on its own position.

If the layer is given a parent later, then the behavior of the layer changes accordingly:

```
idx = index;
if (hasParent) {
    idx = parent.index;
}
thisComp.layer(idx).position.wiggle(5,20)
```

Type

Boolean

25.6 Layer.inPoint

Description

Returns the In point of the layer, in seconds.

Note: In general, the value of outPoint is greater than the value of inPoint. However, if a layer is reversed in time, the value of inPoint is greater than the value of outPoint. Similarly, the value of startTime can be greater than the value of inPoint.

Type

Number

25.7 Layer.outPoint

Description

Returns the Out point of the layer, in seconds.

Type

Number

25.8 Layer.startTime

Description

Returns the start time of the layer, in seconds.

Type

Number

25.9 Layer.hasVideo

Description

Returns `true` if the layer has video, or `false` if it doesn't.

Type

Boolean

25.10 Layer.hasAudio

Description

Returns `true` if the layer has audio or `false` if it doesn't.

Type

Boolean

25.11 Layer.active

Description

Returns `true` if the Video switch is on for the layer and the current time is in the range from the In point of the layer to the Out point of the layer; `false` otherwise.

Type

Boolean

25.12 Layer.enabled

Description

Returns `true` if the Video switch is on for the layer; `false` otherwise.

Type

Boolean

25.13 Layer.audioActive

Description

Returns `true` if the Audio switch is on for the layer and the current time is in the range from the In point of the layer to the Out point of the layer; `false` otherwise.

Type

Boolean

25.14 Layer.sampleImage(point, postEffect=true, t=time) radius=[0.5, 0.5],

Description

Samples the color and alpha channel values of a layer and returns the average alpha-weighted value of the pixels within the specified distance of the point as an array: `[red, green, blue, alpha]`.

If `postEffect` is `true`, the sampled values are for the layer after masks and effects on that layer have been rendered; if `postEffect` is `false`, the sampled values are for the layer before masks and effects have been rendered. The input value `point` is in layer space; the point `[0, 0]` is the center of the upper-left pixel in the layer. The input value `radius` specifies the horizontal and vertical distance from the sample center to the edges of the sampled rectangle. The default value samples one pixel.

This example samples a rectangle 4 pixels wide and 3 pixels high, centered around a point 100 pixels down and to the right of the upper-left corner of the layer:

```
thisComp.layer(1).sampleImage([100, 100], [2, 1.5])
```

Note: The `postEffect` parameter refers to effects applied directly to the layer, not to effects applied indirectly, such as with an adjustment layer.

Note: Using `sampleImage` in an expression no longer disables multiprocessing.

Note: Dan Ebberts provides an example of how to use the `sampleImage` method on his MotionScript website.

Note: Todd Kopriva provides instructions for using the `sampleImage` method and the Point Control effect to monitor colors for a specified point during color correction on his After Effects Region of Interest blog.

Parameters

Argument type: `point` is an Array [2], `radius` is an Array [2], `postEffect` is a Boolean, and `t` is a Number.

Type

Array (4-dimensional)

LAYER PROPERTIES

Description

When you add masks, effects, paint, or text to a layer, After Effects adds new properties to the Timeline panel. There are too many of these properties to list here, so use the pick whip to learn the syntax for referring to them in your expressions.

26.1 Layer.anchorPoint

Description

Returns the anchor point value of the layer in the coordinate system of the layer (layer space).

Type

Array of Numbers (2- or 3-dimensional)

26.2 Layer.position

Description

Returns the position value of the layer, in world space if the layer has no parent. If the layer has a parent, it returns the position value of the layer in the coordinate system of the parent layer (in the layer space of the parent layer).

Type

Array of Numbers (2- or 3-dimensional)

26.3 Layer.scale

Description

Returns the scale value of the layer, expressed as a percentage.

Type

Array of Numbers (2- or 3-dimensional)

26.4 Layer.rotation

Description

Returns the rotation value of the layer in degrees. For a 3D layer, it returns the z rotation value in degrees.

Type

Number

26.5 Layer.opacity

Description

Returns the opacity value for the layer, expressed as a percentage.

Type

Number

26.6 Layer.audioLevels

Description

Returns the value of the Audio Levels property of the layer, in decibels. This value is a 2D value; the first value represents the left audio channel, and the second value represents the right. The value is not the amplitude of the audio track of the source material. Instead, it is the value of the Audio Levels property, which may be affected by keyframes.

Type

Array of Numbers (2-dimensional)

26.7 Layer.timeRemap

Description

Returns the value of the Time Remap property, in seconds, if Time Remap is enabled.

Type

Number

26.8 Layer.marker.key(index)

Description

Returns the MarkerKey object of the layer marker with the specified index.

Parameters

index	Number
-------	--------

Type

MarkerKey

26.9 Layer.marker.key(name)

Description

Returns the MarkerKey object of the layer marker with the specified name. The name value is the name of the marker, as typed in the comment field in the marker dialog box, for example, `marker.key("ch1")`. If more than one marker on the layer has the same name, this method returns the marker that occurs first in time (in layer time). The value for a marker key is a **String**, not a **Number**.

This expression on a property ramps the value of the property from **0** to **100** between two markers identified by name:

```
m1 = marker.key("Start").time;  
m2 = marker.key("End").time;  
linear(time, m1, m2, 0, 100);
```

Parameters

name	String
------	--------

Type

MarkerKey

26.10 Layer.marker.nearestKey(t)

Description

Returns the layer marker that is nearest in time to t.

For example, this expression returns the time of the marker on the layer nearest to the time of 1 second:

```
marker.nearestKey(1).time
```

This expression returns the time of the marker on the layer nearest to the current time:

```
marker.nearestKey(time).time
```

Parameters

t	Number
---	--------

Type

MarkerKey

26.11 Layer.marker.numKeys

Description

Returns the total number of markers on the layer.

Type

Number

26.12 Layer.name

Description

Returns the name of the layer.

Type

String

LAYER SPACE TRANSFORMS

Description

Use layer space transform methods to transform values from one space to another, such as from layer space to world space. The `from` methods transform values from the named space (composition or world) to the layer space.

The `to` methods transform values from the layer space to the named space (composition or world). Each transform method takes an optional argument to determine the time at which the transform is computed; however, you can almost always use the current (default) time.

Use `Vec` transform methods when transforming a direction vector, such as the difference between two position values.

Use the plain (non-`Vec`) transform methods when transforming a point, such as position.

Composition (`comp`) and world space are the same for 2D layers. For 3D layers, however, composition space is relative to the active camera, and world space is independent of the camera.

27.1 `toComp(point, t=time)`

Description

Transforms a point from layer space to composition space.

Parameters

<code>point</code>	Array (2- or 3-dimensional)
<code>t</code>	Number

Type

Array (2- or 3-dimensional)

27.2 fromComp(point, t=time)

Description

Transforms a point from composition space to layer space. The resulting point in a 3D layer may have a nonzero value even though it is in layer space.

Example:

```
fromComp(thisComp.layer(2).position)
```

Parameters

point	Array (2- or 3-dimensional)
t	Number

Type

Array (2- or 3-dimensional)

27.3 toWorld(point, t=time)

Description

Transforms a point from layer space to view-independent world space.

Example:

```
toWorld.effect("Bulge")("Bulge Center")
```

..note:: Dan Ebberts provides an expression on his [MotionScript website](#) that uses the `toWorld` method to auto-orient a layer along only one axis. This is useful, for example, for having characters turn from side to side to follow the camera while remaining upright.

..note:: Rich Young provides a set of expressions on his [AE Portal website](#) that use the `toWorld` method link a camera and light to a layer with the CC Sphere effect.

Parameters

point	Array (2- or 3-dimensional)
t	Number

Type

Array (2- or 3-dimensional)

27.4 fromWorld(point, t=time)

Description

Transforms a point from world space to layer space.

Example:

```
fromWorld(thisComp.layer(2).position)
```

See Expression example: Create a bulge between two layers for an example of how this method can be used.

Parameters

point	Array (2- or 3-dimensional)
t	Number

Type

Array (2- or 3-dimensional)

27.5 toCompVec(vec, t=time)

Description

Transforms a vector from layer space to composition space.

Example:

```
toCompVec([1, 0])
```

Parameters

vec	Array (2- or 3-dimensional)
t	Number

Type

Array (2- or 3-dimensional)

27.6 fromCompVec(vec, t=time)

Description

Transforms a vector from composition space to layer space.

Example (2D layer):

```
dir = sub(position, thisComp.layer(2).position);
fromCompVec(dir)
```

Parameters

vec	Array (2- or 3-dimensional)
t	Number

Type

Array (2- or 3-dimensional)

27.7 toWorldVec(vec, t=time)

Description

Transforms a vector from layer space to world space.

Example:

```
p1 = effect("Eye Bulge 1")("Bulge Center");
p2 = effect("Eye Bulge 2")("Bulge Center");
toWorld(sub(p1, p2))
```

Parameters

vec	Array (2- or 3-dimensional)
t	Number

Type

Array (2- or 3-dimensional)

27.8 fromWorldVec(vec, t=time)

Description

Transforms a vector from world space to layer space.

Example:

```
fromWorld(thisComp.layer(2).position)
```

Parameters

vec	Array (2- or 3-dimensional)
t	Number

Type

Array (2- or 3-dimensional)

27.9 fromCompToSurface(point, t=time)

Description

Projects a point located in composition space to a point on the surface of the layer (zero z-value) at the location where it appears when viewed from the active camera. This method is useful for setting effect control points. Use with 3D layers only.

Parameters

point	Array (2- or 3-dimensional)
t	Number

Type

Array (2-dimensional)

28.1 Layer.orientation

Description

Returns the 3D orientation value, in degrees, for a 3D layer.

Type

Array (3-dimensional)

28.2 Layer.rotationX

Description

Returns the x rotation value, in degrees, for a 3D layer.

Type

Number

28.3 Layer.rotationY

Description

Returns the y rotation value, in degrees, for a 3D layer.

Type

Number

28.4 Layer.rotationZ

Description

Returns the z rotation value, in degrees, for a 3D layer.

Type

Number

28.5 Layer.lightTransmission

Description

Returns the value of the Light Transmission property for a 3D layer.

Type

Number

28.6 Layer.castsShadows

Description

Returns a value of 1 if the layer casts shadows and 2 if the property is set to `Only`.

Type

Number

28.7 Layer.acceptsShadows

Description

Returns a value of 1 if the layer accepts shadows and 2 if the property is set to `Only`.

Type

Number

28.8 Layer.acceptsLights

Description

Returns a value of 1 if the layer accepts lights.

Type

Boolean Number

28.9 Layer.ambient

Description

Returns the ambient component value as a percentage.

Type

Number

28.10 Layer.diffuse

Description

Returns the diffuse component value as a percentage.

Type

Number

28.11 Layer.specular

Description

Returns the specular component value as a percentage.

Type

Number

28.12 Layer.shininess

Description

Returns the shininess component value as a percentage.

Type

Number

28.13 Layer.metal

Description

Returns the metal component value as a percentage.

Type

Number

This category holds generic text-related entries for text layers.

29.1 Text.sourceText

Description

Returns the text content of a text layer.

As of After Effects 17.0, this property returns the *Source Text* object to access text style properties. If no style properties are specified, this returns the text content as expected.

Type

String of text content, or *Source Text* (AE 17.0+)

29.2 Text.Font...

Description

Launches a dialog window for the user to specify a font name and weight.

Upon selection, the internal font name is injected into the expression editor as a string.

Type

String

SOURCE TEXT

These functions are accessible from *Text.sourceText* after AE 17.0.

30.1 SourceText.style

Description

Returns the text style object for a given `sourceText` property.

Type

Text Style object

30.2 SourceText.getStyleAt(charIndex, t = time)

Description

This function returns the style value of a particular character at a specific time.

In case the style is keyframed and changes over time, use the second `time` parameter to specify the target time to get the style at.

Note: Using *SourceText.style* is the same as using `text.sourceText.getStyleAt(0,0)`

For example, to get the style of the first character at the beginning of the timeline:

```
text.sourceText.getStyleAt(0,0);
```

Parameters

<code>index</code>	Number. The index of the letter or character whose style is needed
<code>time</code>	Number, optional. The time within the composition to get the style from. Defaults to <code>time</code> .

Type

Text Style object

30.3 SourceText.createStyle()

Description

Used to initialize an empty *Text Style* object in which you'd manually bake in specific values.

For example, to create a new style with font size 300 and the font Impact:

```
text.sourceText.createStyle().setFontSize(300).setFont("Impact");
```

Parameters

None.

Type

Empty *Text Style* object

TEXT STYLE

These functions are accessible from *SourceText.style* after AE 17.0.

For more info on working with text styles, see:

- [Use Expressions to Edit and Access Text Properties on helpx.adobe.com](https://helpx.adobe.com/after-effects/using/expressions-to-edit-and-access-text-properties.html)
- [After Effects 2020: Express Yourself \(and Your Text\) on blog.adobe.com](https://blog.adobe.com/en/post/2020/07/after-effects-2020-express-yourself-and-your-text/)

31.1 TextStyle.setText(value)

Description

This is used when you want to define (or inherit) a text style, while setting the text content separately.

For example, to inherit the text style and content from another layer:

```
const referenceText = thisComp.layer("source layer name").text.sourceText;  
  
const style = referenceText.getStyleAt(0,0);  
style.setText(referenceText);
```

Or to create a custom style and then set the text within the expression:

```
text.sourceText.createStyle().setFontSize(300).setFont("Impact").setText("Hello world!")
```

Parameters

value	String. The text to set
-------	-------------------------

Type

None

31.2 TextStyle.fontSize

Description

Returns the value of Font Size for a text layer.

Type

Number

31.3 TextStyle.setFontSize(value)

Description

Used to set the font size to a specified value.

Parameters

value	Number. The font size to set to.
-------	----------------------------------

Type

None

31.4 TextStyle.font

Description

Returns the font name for a text layer.

Type

String

31.5 TextStyle.setFont(value)

Description

Used to set the font to a specified value.

Parameters

value	String. The font to set to.
-------	-----------------------------

Type

None

31.6 TextStyle.isFauxBold

Description

Returns whether faux bold is enabled

Type

Boolean

31.7 TextStyle.setFauxBold(value)

Description

Used to set the faux bold status

Parameters

value	Boolean. Whether to enable or disable faux bold
-------	---

Type

None

31.8 TextStyle.isFauxItalic

Description

Returns whether faux italic is enabled

Type

Boolean

31.9 TextStyle.setFauxItalic(value)

Description

Used to set the faux italic status

Parameters

value	Boolean. Whether to enable or disable faux italic
-------	---

Type

None

31.10 TextStyle.isAllCaps

Description

Returns whether all caps is enabled

Type

Boolean

31.11 TextStyle.setAllCaps(value)

Description

Used to set the All Caps status

Parameters

value	Boolean. Whether to enable or disable all caps
-------	--

Type

None

31.12 TextStyle.isSmallCaps

Description

Returns whether small caps is enabled

Type

Boolean

31.13 TextStyle.setSmallCaps(value)

Description

Used to set the small caps status

Parameters

value	Boolean. Whether to enable or disable small caps
-------	--

Type

None

31.14 TextStyle.tracking

Description

Returns the value of Tracking for a text layer.

Type

Number

31.15 TextStyle.setTracking(value)

Description

Used to set the tracking to a specified value.

Parameters

value	Number. The tracking value to set to.
-------	---------------------------------------

Type

None

31.16 TextStyle.leading

Description

Returns the value of leading for a text layer.

Type

Number

31.17 TextStyle.setLeading(value)

Description

Used to set the leading to a specified value.

Parameters

value	Number. The leading value to set to.
-------	--------------------------------------

Type

None

31.18 TextStyle.isAutoLeading

Description

Returns whether auto leading is enabled

Type

Boolean

31.19 TextStyle.setAutoLeading(value)

Description

Used to set the auto leading status

Parameters

value	Boolean. Whether to enable or disable auto leading
-------	--

Type

None

31.20 TextStyle.baselineShift

Description

Returns the value of baseline shift for a text layer.

Type

Number

31.21 TextStyle.setBaselineShift(value)

Description

Used to set the baseline shift to a specified value.

Parameters

value	Number. The baseline shift value to set to.
-------	---

Type

None

31.22 TextStyle.applyFill

Description

Returns whether text fill color is enabled

Type

Boolean

31.23 TextStyle.setApplyFill(value)

Description

Used to set whether text fill is enabled

Parameters

value	Boolean. Whether to enable or disable apply fill
-------	--

Type

None

31.24 TextStyle.fillColor

Description

Returns the text fill color, RGB values on a scale from 0 - 1.0

Type

Array of numbers

31.25 TextStyle.setFillColor(value)

Description

Used to set the text fill color

Parameters

value	Array of numbers. [R, G, B] with each value between 0.0 to 1.0
-------	--

Type

None

31.26 TextStyle.applyStroke

Description

Returns whether text stroke is enabled

Type

Boolean

31.27 TextStyle.setApplyStroke(value)

Description

Used to set whether text stroke is enabled

Parameters

value	Boolean. Whether to enable or disable text stroke
-------	---

Type

None

31.28 TextStyle.strokeColor

Description

Returns the text stroke color, RGB values on a scale from 0 - 1.0

Type

Array of numbers

31.29 TextStyle.setStrokeColor(value)

Description

Used to set the text stroke color

Parameters

value	Array of numbers. [R, G, B] with each value between 0.0 to 1.0
-------	--

Type

None

31.30 TextStyle.strokeWidth

Description

Returns the stroke width value for a text layer.

Type

Number

31.31 TextStyle.setStrokeWidth(value)

Description

Used to set the stroke width to a specified value.

Parameters

value	Number. The value to set the stroke width to.
-------	---

Type

None