Nomenclature

Tweener A tween that takes control of a value and animates it.

Sequence A special tween that, instead of taking control of a value, takes control of other tweens and animates them as a group.

Tween A generic word that indicates both a Tweener and a Sequence.

Nested tween A tween contained inside a Sequence.

Prefixes

Prefixes are important to use the most out of **IntelliSense**, so try to remember these:

DO Prefix for all tween shortcuts (operations that can be started directly from a known object, like a transform or a material). Also the prefix of the main DOTween class.

```
transform.DOMoveX(100, 1);
transform.DORestart();
DOTween.Play();
```

Set Prefix for all settings that can be chained to a tween (except for From, since it's applied as a setting but is not really a setting).

```
myTween.SetLoops(4, LoopType.Yoyo).SetSpeedBased();
```

On Prefix for all callbacks that can be chained to a tween.

```
myTween.OnStart(myStartFunction).OnComplete(myCompleteFunction);
```

DOTween.Init

The first time you create a tween, DOTween will initialize itself automatically, using default values.

If instead you prefer to initialize it yourself (recommended), call this methods once, BEFORE creating any tween (calling it afterwards will have no effect).

Consider that you can still change all init settings whenever your want, by using DOTween's global settings

Optionally, you can chain `SetCapacity` to the Init method, which allows to set the max Tweeners/Sequences initial capacity (it's the same as calling DOTween.SetTweensCapacity later).

Expand all

```
static DOTween.Init(bool recycleAllByDefault = false, bool useSafeMode = true, LogBehaviour
logBehaviour = LogBehaviour.ErrorsOnly)
```

```
// EXAMPLE A: initialize with the preferences set in DOTween's Utility Panel
DOTween.Init();
// EXAMPLE B: initialize with custom settings, and set capacities immediately
DOTween.Init(true, true, LogBehaviour.Verbose).SetCapacity(200, 10);
```

TO TOP

Creating a Tweener

Tweeners are the working ants of DOTween. They take a property/field and animate it towards a given value.

As of now DOTween can tween these types of values:
float, double, int, uint, long, ulong, Vector2/3/4, Quaternion, Rect, RectOffset, Color, string
(some of these values can be tweened in **special ways**)

Also, you can create **custom DOTween plugins** to tween custom value types

There are 3 ways to create a Tweener: the generic way, the shortcuts way and additional generic ways.

A. The generic way

This is the most flexible way of tweening and allows you to tween almost any value, either public or private, static or dynamic (just so you know, the shortcuts way actually uses the generic way in the background).

As with shortcuts, the generic way has a FROM alternate version. Just chain a From to a Tweener to make the tween behave as a FROM tween instead of a TO tween.

Expand all

```
static DOTween.To(getter, setter, to, float duration)
```

B. The shortcuts way

DOTween includes shortcuts for some known Unity objects, like Transform, Rigidbody and Material. You can start a tween directly from a reference to these objects (which will also automatically set the object itself as the tween target), like:

```
transform.DOMove(new Vector3(2,3,4), 1);
rigidbody.DOMove(new Vector3(2,3,4), 1);
material.DOColor(Color.green, 1);
```

Each of these shortcuts also has a FROM alternate version except where indicated. Just chain a From to a Tweener to make the tween behave as a FROM tween instead of a TO tween.
IMPORTANT: when you assign a FROM to a tween, the target will immediately jump to the FROM position (immediately as in "the moment you write that line of code", not "the moment the tween starts").

```
transform.DOMove(new Vector3(2,3,4), 1).From();
rigidbody.DOMove(new Vector3(2,3,4), 1).From();
material.DOColor(Color.green, 1).From();
```

Basic elements shortcuts

AudioMixer (Unity 5)

Expand all

```
DOSetFloat(string floatName, float to, float duration)
```

AudioSource

Expand all

```
DOFade(float to, float duration)
```

DOPitch(float to, float duration)

Camera

Expand all

DOAspect(float to, float duration)
DOColor(Color to, float duration)
DOFarClipPlane(float to, float duration)
DOFieldOfView(float to, float duration)
DONearClipPlane(float to, float duration)
DOOrthoSize(float to, float duration)
DOPixelRect(Rect to, float duration)
DORect(Rect to, float duration)
DOShakePosition(float duration, float/Vector3 strength, int vibrato, bool fadeOut)
DOShakeRotation(float duration, float/Vector3 strength, int vibrato, float randomness bool fadeOut)

Light

Expand all

DOColor(Color to, float duration)
DOIntensity(float to, float duration)
DOShadowStrength(float to, float duration)

Blendable tweens

DOBlendableColor(Color to, float duration)

LineRenderer

Expand all

DOColor(Color2 startValue, Color2 endValue, float duration)

Material

Expand all

DOColor(Color to, float duration)
DOColor(Color to, string property, float duration)
DOFade(float to, float duration)
DOFade(float to, string property, float duration)
DOFloat(float to, string property, float duration)
DOGradientColor(Gradient to, float duration)
DOGradientColor(Gradient to, string property, float duration)
DOOffset(Vector2 to, float duration)
DOOffset(Vector2 to, string property, float duration)
DOTiling(Vector2 to, float duration)
DOTiling(Vector2 to, string property, float duration)
DOVector(Vector4 to, string property, float duration)

Blendable tweens

DOBlendableColor(Color to, float duration)
DOBlendableColor(Color to, string property, float duration)

Rigidbody

These shortcuts use rigidbody's MovePosition/MoveRotation methods in the background, to correctly animate things related to physics objects.

Expand all

Move

DOMove(Vector3 to, float duration, bool snapping)
DOMoveX/DOMoveY/DOMoveZ(float to, float duration, bool snapping)
DOJump(Vector3 endValue, float jumpPower, int numJumps, float duration, bool snapping)

Rotate

DORotate(Vector3 to, float duration, RotateMode mode)
DOLookAt(Vector3 towards, float duration, AxisConstraint axisConstraint = AxisConstraint.None, Vector3 up = Vector3.up)

PRO ONLY ➡ Spiral - no FROM

DOSpiral(float duration, Vector3 axis = null, SpiralMode mode = SpiralMode.Expand, float speed = 1, float frequency = 10, float depth = 0, bool snapping = false)

Rigidbody2D

These shortcuts use rigidbody2D's MovePosition/MoveRotation methods in the background, to correctly animate things related to physics objects.

Expand all

Move

DOMove(Vector2 to, float duration, bool snapping)
DOMoveX/DOMoveY(float to, float duration, bool snapping)
DOJump(Vector2 endValue, float jumpPower, int numJumps, float duration, bool snapping)

Rotate

DORotate(float toAngle, float duration)

SpriteRenderer

Expand all

DOColor(Color to, float duration)
DOFade(float to, float duration)
DOGradientColor(Gradient to, float duration)

Blendable tweens

DOBlendableColor(Color to, float duration)

TrailRenderer

Expand all

DOResize(float toStartWidth, float toEndWidth, float duration)
DOTime(float to, float duration)

Transform

Expand all

Move

DOMove(Vector3 to, float duration, bool snapping)
DOMoveX/DOMoveY/DOMoveZ(float to, float duration, bool snapping)
DOLocalMove(Vector3 to, float duration, bool snapping)
DOLocalMoveX/DOLocalMoveY/DOLocalMoveZ(float to, float duration, bool snapping)
DOJump(Vector3 endValue, float jumpPower, int numJumps, float duration, bool snapping)
DOLocalJump(Vector3 endValue, float jumpPower, int numJumps, float duration, bool snapping)

Rotate

DORotate(Vector3 to, float duration, RotateMode mode)
DORotateQuaternion(Quaternion to, float duration)
DOLocalRotate(Vector3 to, float duration, RotateMode mode)
DOLocalRotateQuaternion(Quaternion to, float duration)
DOLookAt(Vector3 towards, float duration, AxisConstraint axisConstraint = AxisConstraint.None, Vector3 up = Vector3.up)

Scale

DOScale(float/Vector3 to, float duration)
DOScaleX/DOScaleY/DOScaleZ(float to, float duration)

Punch – no FROM

DOPunchPosition(Vector3 punch, float duration, int vibrato, float elasticity, bool snapping)
DOPunchRotation(Vector3 punch, float duration, int vibrato, float elasticity)
DOPunchScale(Vector3 punch, float duration, int vibrato, float elasticity)

Shake – no FROM

DOShakePosition(float duration, float/Vector3 strength, int vibrato, float randomness, bool snapping, bool fadeOut)
DOShakeRotation(float duration, float/Vector3 strength, int vibrato, float randomness, bool fadeOut)
DOShakeScale(float duration, float/Vector3 strength, int vibrato, float randomness, bool fadeOut)

Path – no FROM

DOPath(Vector3[] waypoints, float duration, PathType pathType = Linear, PathMode pathMode = Full3D, int resolution = 10, Color gizmoColor = null)
DOLocalPath(Vector3[] waypoints, float duration, PathType pathType = Linear, PathMode pathMode = Full3D, int resolution = 10, Color gizmoColor = null)

Blendable tweens

DOBlendableMoveBy(Vector3 by, float duration, bool snapping)
DOBlendableLocalMoveBy(Vector3 by, float duration, bool snapping)
DOBlendableRotateBy(Vector3 by, float duration, RotateMode mode)
DOBlendableLocalRotateBy(Vector3 by, float duration, RotateMode mode)
DOBlendableScaleBy(Vector3 by, float duration)

PRO ONLY ➡ Spiral – no FROM

DOSpiral(float duration, Vector3 axis = null, SpiralMode mode = SpiralMode.Expand, float speed = 1, float frequency = 10, float depth = 0, bool snapping = false)

Tween

These are shortcuts that actually tween other tweens properties. I bet you didn't think you could do it :P

Expand all

DOTimeScale(float toTimeScale, float duration)

Unity UI 4.6 shortcuts

CanvasGroup (Unity UI 4.6)

Expand all

```
DOFade(float to, float duration)
```

Graphic (Unity UI 4.6)

Expand all

```
DOColor(Color to, float duration)
DOFade(float to, float duration)
```

Blendable tweens

```
DOBlendableColor(Color to, float duration)
```

Image (Unity UI 4.6)

Expand all

```
DOColor(Color to, float duration)
DOFade(float to, float duration)
DOFillAmount(float to, float duration)
DOGradientColor(Gradient to, float duration)
```

Blendable tweens

```
DOBlendableColor(Color to, float duration)
```

LayoutElement (Unity UI 4.6)

Expand all

```
DOFlexibleSize(Vector2 to, float duration, bool snapping)
DOMinSize(Vector2 to, float duration, bool snapping)
DOPreferredSize(Vector2 to, float duration, bool snapping)
```

Outline (Unity UI 4.6)

Expand all

```
DOColor(Color to, float duration)
DOFade(float to, float duration)
```

RectTransform (Unity UI 4.6)

Expand all

```
DOAnchorMax(Vector2 to, float duration, bool snapping)
DOAnchorMin(Vector2 to, float duration, bool snapping)
DOAnchorPos(Vector2 to, float duration, bool snapping)
DOAnchorPosX/DOAnchorPosY(float to, float duration, bool snapping)
DOAnchorPos3D(Vector3 to, float duration, bool snapping)
DOJumpAnchorPos(Vector2 endValue, float jumpPower, int numJumps, float duration, bool
snapping)
DOPivot(Vector2 to, float duration)
DOPivotX/DOPivotY(float to, float duration)
DOPunchAnchorPos(Vector2 punch, float duration, int vibrato, float elasticity, bool
snapping)
DOShakeAnchorPos(float duration, float/Vector3 strength, int vibrato, float randomness, bool
snapping, bool fadeOut)
DOSizeDelta(Vector2 to, float duration, bool snapping)
```

ScrollRect (Unity UI 4.6)

Expand all

```
DONormalizedPos(Vector2 to, float duration, bool snapping)
DOHorizontalNormalizedPos(float to, float duration, bool snapping)
DOVerticalPos(float to, float duration, bool snapping)
```

Slider (Unity UI 4.6)

Expand all

```
DOValue(float to, float duration, bool snapping = false)
```

Text (Unity UI 4.6)

Expand all

```
DOColor(Color to, float duration)
DOFade(float to, float duration)
DOText(string to, float duration, bool richTextEnabled = true, ScrambleMode scrambleMode =
ScrambleMode.None, string scrambleChars = null)
```

Blendable tweens

DOBlendableColor(Color to, float duration)

PRO ONLY ➡ 2D Toolkit shortcuts

tk2dBaseSprite

Expand all

DOScale(Vector3 to, float duration)
DOScaleX/Y/Z(float to, float duration)
DOColor(Color to, float duration)
DOFade(float to, float duration)

tk2dSlicedSprite

Expand all

DOScale(Vector2 to, float duration)
DOScaleX/Y(float to, float duration)

tk2dTextMesh

Expand all

DOColor(Color to, float duration)
DOFade(float to, float duration)
DOText(string to, float duration, bool richTextEnabled = true, ScrambleMode scrambleMode = ScrambleMode.None, string scrambleChars = null)

PRO ONLY ➡ TextMesh Pro shortcuts

TextMeshPro + TextMeshProUGUI

Expand all

DOScale(float to, float duration)
DOColor(Color to, float duration)
DOFaceColor(Color to, float duration)
DOFaceFade(float to, float duration)
DOFade(float to, float duration)
DOFontSize(float to, float duration)
DOGlowColor(Color to, float duration)
DOMaxVisibleCharacters(int to, float duration)
DOOutlineColor(Color to, float duration)
DOText(string to, float duration, bool richTextEnabled = true, ScrambleMode scrambleMode = ScrambleMode.None, string scrambleChars = null)

C. Additional generic ways

These are additional generic methods that allow to tween values in specific ways.

These too have FROM alternate versions except where indicated. Just chain a From to a Tweener to make the tween behave as a FROM tween instead of a TO tween.

Expand all

static DOTween.Punch(getter, setter, Vector3 direction, float duration, int vibrato, float elasticity)
static DOTween.Shake(getter, setter, float duration, float/Vector3 strength, int vibrato, float randomness, bool ignoreZAxis)
static DOTween.ToAlpha(getter, setter, float to, float duration)
static DOTween.ToArray(getter, setter, float to, float duration)
static DOTween.ToAxis(getter, setter, float to, float duration, AxisConstraint axis)

Virtual Tween

static DOTween.To(setter, float startValue, float endValue, float duration)

TO TOP

Creating a Sequence

Sequences are like Tweeners, but instead of animating a property or value they animate other Tweeners or Sequences as a group.

Sequences can be contained inside other Sequences without any limit to the depth of the hierarchy.

The sequenced tweens don't have to be one after each other. You can overlap tweens with the Insert method.

A tween (Sequence or Tweener) can be nested only inside a single other Sequence, meaning you can't reuse the same tween in multiple Sequences. Also, the main Sequence will take control of all its nested elements, and you won't be able to control nested tweens separately (consider the Sequence as a movie timeline which becomes fixed once it starts up the for the very first time).

IMPORTANT: don't use empty Sequences.

To create a Sequence, you follow these two steps:

1. Grab a new Sequence to use and store it as a reference

Expand all

```
static DOTween.Sequence()
```

2. Add tweens, intervals and callbacks to your Sequence

Note that all these methods need to be applied before the Sequence starts (usually the next frame after you create it, unless it's paused), or they won't have any effect.

Also note that any nested Tweener/Sequence needs to be fully created before adding it to a Sequence, because after that it will be locked.

Delays and loops (when not infinite) will work even inside nested tweens.

Expand all

```
Append(Tween tween)
AppendCallback(TweenCallback callback)
AppendInterval(float interval)
Insert(float atPosition, Tween tween)
InsertCallback(float atPosition, TweenCallback callback)
Join(Tween tween)
Prepend(Tween tween)
PrependCallback(TweenCallback callback)
PrependInterval(float interval)
```

TIP: You can create Sequences made only of callbacks and use them as timers or stuff like that.

Examples

Creating a Sequence

```
// Grab a free Sequence to use
Sequence mySequence = DOTween.Sequence();
// Add a movement tween at the beginning
mySequence.Append(transform.DOMoveX(45, 1));
// Add a rotation tween as soon as the previous one is finished
mySequence.Append(transform.DORotate(new Vector3(0,180,0), 1));
// Delay the whole Sequence by 1 second
mySequence.PrependInterval(1);
// Insert a scale tween for the whole duration of the Sequence
mySequence.Insert(0, transform.DOScale(new Vector3(3,3,3), mySequence.Duration()));
```

Same as the previous example but with chaining (plus line breaks to make things clearer):

```
Sequence mySequence = DOTween.Sequence();
mySequence.Append(transform.DOMoveX(45, 1))
  .Append(transform.DORotate(new Vector3(0,180,0), 1))
  .PrependInterval(1)
  .Insert(0, transform.DOScale(new Vector3(3,3,3), mySequence.Duration()));
```

TO TOP

Settings, options and callbacks

DOTween uses a chaining approach when it comes to applying settings to a tween. Or you can change the global default options that will be applied to all newly created tweens.

Global settings

General settings

Expand all

```
static LogBehaviour DOTween.logBehaviour
static bool DOTween.maxSmoothUnscaledTime
static bool DOTween.showUnityEditorReport
static float DOTween.timeScale
static bool DOTween.useSafeMode
static bool DOTween.useSmoothDeltaTime
static DOTween.SetTweensCapacity(int maxTweeners, int maxSequences)
```

Settings applied to all newly created tweens

Expand all

```
static bool DOTween.defaultAutoKill
static AutoPlay DOTween.defaultAutoPlay
static float DOTween.defaultEaseOvershootOrAmplitude
static float DOTween.defaultEasePeriod
static Ease DOTween.defaultEaseType
static LoopType DOTween.defaultLoopType
static bool DOTween.defaultRecyclable
static bool DOTween.defaultTimeScaleIndependent
static UpdateType DOTween.defaultUpdateType
```

Tweener and Sequence settings

Instance properties

Expand all

```
float timeScale
```

Chained settings

These settings can be chained to all types of tweens.
You can also chain them while a tween is running (except for **SetLoops** and **SetAs**)

Expand all

```
SetAs(Tween tween \ TweenParams tweenParams)
SetAutoKill(bool autoKillOnCompletion = true)
SetEase(Ease easeType \ AnimationCurve animCurve \ EaseFunction customEase)
SetId(object id)
SetLoops(int loops, LoopType loopType = LoopType.Restart)
SetRecyclable(bool recyclable)
SetRelative(bool isRelative = true)
SetUpdate(UpdateType updateType, bool isIndependentUpdate = false)
```

Chained callbacks

Expand all

```
OnComplete(TweenCallback callback)
OnKill(TweenCallback callback)
OnPlay(TweenCallback callback)
OnPause(TweenCallback callback)
OnRewind(TweenCallback callback)
OnStart(TweenCallback callback)
OnStepComplete(TweenCallback callback)
OnUpdate(TweenCallback callback)
OnWaypointChange(TweenCallback<int> callback)
```

By the way, callbacks attached to nested tweens will still work in the correct order.

If you want to use a callback with parameters, lambdas come to the rescue:

```
// Callback without parameters
transform.DOMoveX(4, 1).OnComplete(MyCallback);
// Callback with parameters
transform.DOMoveX(4, 1).OnComplete(()=>MyCallback(someParam, someOtherParam));
```

Tweener-specific settings and options

These settings are specific to Tweeners, and will have no effect on Sequences.
Apart from **SetEase**, chaining these settings while a tween is running will have no effect.

Expand all

```
From(bool isRelative = false)
SetDelay(float delay)
SetSpeedBased(bool isSpeedBased = true)
```

SetOptions

Some Tweeners have specific special options that will be available to you depending on the type of thing you're tweening. It's all automatic: if a Tweener has specific options you'll see a specific **SetOptions** methods present for that Tweener, otherwise you won't. It's magic!

Note that these options are usually available only when creating tweens via the generic way, while shortcuts have the same options already included in their main creation method.

The important thing to remember is that, while all other settings can be chained together in any order, **SetOptions** must be chained immediately after the tween creation function, or it won't be available anymore.

Expand all

Generic Tweens Specific Options (already included in the corresponding tween shortcuts)

```
Color tween ➡ SetOptions(bool alphaOnly)
float tween ➡ SetOptions(bool snapping)
Quaternion tween ➡ SetOptions(bool useShortest360Route)
Rect tween ➡ SetOptions(bool snapping)
String tween ➡ SetOptions(bool richTextEnabled, ScrambleMode scrambleMode =
ScrambleMode.None, string scrambleChars = null)
Vector2/3/4 tween ➡ SetOptions(AxisConstraint constraint, bool snapping)
Vector3Array tween ➡ SetOptions(bool snapping)
```

DOPath Specific Options

```
Path tween ➡ SetOptions(bool closePath, AxisConstraint lockPosition = AxisConstraint.None,
AxisConstraint lockRotation = AxisConstraint.None)
Path tween ➡ SetLookAt(Vector3 lookAtPosition/lookAtTarget/lookAhead, Vector3
forwardDirection, Vector3 up = Vector3.up)
```

TweenParams

If you used HOTween previously, you will know TweenParms (now called TweenParams): they're used to store settings that you can then apply to multiple tweens. The difference from HOTween is that they're not necessary at all, since now settings chaining is done directly on a tween. Instead they're here only as an extra utility class.

To use it, create a new TweenParams instance or **Clear()** an existing one, then add settings like you do with regular tween chaining. To apply them to a tween you then use **SetAs**.

```
// Store settings for an infinite looping tween with elastic ease
TweenParams tParms = new TweenParams().SetLoops(-1).SetEase(Ease.OutElastic);
// Apply them to a couple of tweens
transformA.DOMoveX(15, 1).SetAs(tParms);
transformB.DOMoveY(10, 1).SetAs(tParms);
```

More on chaining

Just to be clear, you don't need to chain one thing at a time and you can do this instead:

```
transform.DOMoveX(45, 1).SetDelay(2).SetEase(Ease.OutQuad).OnComplete(MyCallback);
```

TO TOP

Controlling a tween

You have 3 ways to manipulate a tween. All of them share the same method names, except for shortcut-enhanced ones which have an additional **DO** prefix.

A. Via static methods and filters

The DOTween class contains many static methods that allow you to control tweens. Each of them comes both in an "All" version (like `DOTween.KillAll`), which applies to all existing tweens, and a simple version (`DOTween.Kill(myTargetOrId)`) with a parameter that allows you to filter operations by a tween's id or target (id is set manually via SetId, while target is set automatically when creating a tween via a shortcut).

Static methods additionally return an int, which represents all the tweens that were actually able to perform the requested operation.

```
// Pauses all tweens
DOTween.PauseAll();
// Pauses all tweens that have "badoom" as an id
DOTween.Pause("badoom");
// Pauses all tweens that have someTransform as a target
DOTween.Pause(someTransform);
```

B. Directly from the tween

instead of using static methods you can just call the same methods from your tween's reference.

```
myTween.Pause();
```

C. From a shortcut-enhanced reference

Same as above, but you can call those same methods from a shortcut-enhanced object. Remember that in this case the method names have an additional `DO` prefix to distinguish them from the regular object methods.

```
transform.DOPause();
```

Control methods

Again, remember that all these method names are shared by all manipulation ways, but that in case of object shortcuts there's an additional `DO` prefix.

IMPORTANT: remember that to use these methods on a tween after it has ended, you have to disable its autoKill behaviour, otherwise a tween is automatically killed at completion.

Expand all

```
CompleteAll/Complete(bool withCallbacks = false)
FlipAll/Flip()
GotoAll/Goto(float to, bool andPlay = false)
KillAll/Kill(bool complete = true, params object[] idsOrTargetsToExclude)
PauseAll/Pause()
PlayAll/Play()
PlayBackwardsAll/PlayBackwards()
PlayForwardAll/PlayForward()
RestartAll/Restart(bool includeDelay = true, float changeDelayTo = -1)
RewindAll/Rewind(bool includeDelay = true)
SmoothRewindAll/SmoothRewind()
TogglePauseAll/TogglePause()
```

Special control methods

Common for all tweens

Expand all

```
ForceInit()
```

Type-specific

These are special control methods that work only on some specific type of tweens

Expand all

```
GotoWaypoint(int waypointIndex, bool andPlay = false)
```

TO TOP

Getting data from tweens

Static methods (DOTween)

Expand all

```
static List<Tween> PausedTweens()
static List<Tween> PlayingTweens()
static List<Tween> TweensById(object id, bool playingOnly = false)
static List<Tween> TweensByTarget(object target, bool playingOnly = false)
static bool IsTweening(object idOrTarget, bool alsoCheckIfPlaying = false)
static int TotalPlayingTweens()
```

Instance methods (Tween/Tweener/Sequence)

Expand all

```
float fullPosition
int CompletedLoops()
float Delay()
float Duration(bool includeLoops = true)
float Elapsed(bool includeLoops = true)
float ElapsedDirectionalPercentage()
float ElapsedPercentage(bool includeLoops = true)
bool IsActive()
bool IsBackwards()
bool IsComplete()
bool IsInitialized()
bool IsPlaying()
int Loops()
```

Instance methods ➡ Path tweens

Expand all

```
Vector3 PathGetPoint(float pathPercentage)
Vector3[] PathGetDrawPoints(int subdivisionsXSegment = 10)
float PathLength()
```

TO TOP

WaitFor coroutines

Tweens come with a useful set of YieldInstructions which you can place inside your Coroutines, and that allow you to wait for something to happen.

Expand all

```
WaitForCompletion()
WaitForElapsedLoops(int elapsedLoops)
WaitForKill()
WaitForPosition(float position)
WaitForRewind()
WaitForStart()
```

TO TOP

Additional methods

Static methods (DOTween)

Expand all

```
static DOTween.Clear(bool destroy = false)
static DOTween.ClearCachedTweens()
static DOTween.Validate()
static DOTween.ManualUpdate(float deltaTime, float unscaledDeltaTime)
```

Instance methods (Tween/Tweener/Sequence)

Expand all

Tweener

```
ChangeEndValue(newEndValue, float duration = -1, bool snapStartValue = false)
ChangeStartValue(newStartValue, float duration = -1)
ChangeValues(newStartValue, newEndValue, float duration = -1)
```

TO TOP

Virtual methods

NOTE: virtual methods can't be placed inside Sequences.

Expand all

```
static Tweener DOVirtual.Float(float from, float to, float duration, TweenCallback<float>
onVirtualUpdate)
static float DOVirtual.EasedValue(float from, float to, float lifetimePercentage, Ease
easeType \ AnimationCurve animCurve)
static Tween DOVirtual.DelayedCall(float delay, TweenCallback callback, bool
ignoreTimeScale = true)
```

TO TOP

Creating custom plugins

The sample UnityPackage from the examples page shows, among other things, how to create custom plugins.

TO TOP