

# Ai→Canvas

A project by [Mike Swanson](#)

This extended documentation applies to version 1.2 of the plug-in.

## Installation

Version 1.2 of the Ai→Canvas plug-in works with Adobe® Illustrator® CS6 and CC on both Mac and PC. To install, copy the correct *.aip* file to the Illustrator plug-ins folder. For a default PC installation, this should be something like *C:\Program Files\Adobe\Adobe Illustrator CC (64 Bit)\Plug-ins*. On a Mac, it should be something like */Applications/Adobe Illustrator CC/Plug-ins*.

The next time Illustrator is started, pull down the *File* menu, select *Export...*, and verify that “<canvas> (\*.html)” is available in the *Save as type* drop-down list. There are no other options to configure or runtimes to install.

After an export, the plug-in automatically launches the exported file. If *.html* files have been associated with a browser, the exported file is displayed. Otherwise, it is opened in the application that is associated with the *.html* file type.

## Layers

For drawings that don’t require any special functionality, simply export the artwork normally. All of the layers will be exported as canvas drawing commands that are included in a single JavaScript function called *draw*.

To take advantage of additional Ai→Canvas functionality, name a layer like a JavaScript function. Most importantly, make sure that it starts with a valid name, includes an opening parenthesis, and ends with a closing parenthesis and semicolon. For example:

```
drawShape();
```

To add a property setting, include the property name, a colon, and its value between the parentheses. Multiple properties are separated by semicolons, and they can be in any order. For example:

```
drawShape(origin: center; rasterize: shape);
```

## Property Names

Each property has both long and short names (for convenience). Many values also have short versions. Long and short names can be used in any combination. In the property lists in this document, default

property values appear in **bold**, and short equivalents are listed after the slash. For example, these two layer names have the same effect:

- `drawShape(origin: center; rasterize: shape);`
- `drawShape(o:c;rast:shape);`

## Layer Types

A layer can be one of two types:

1. **Drawing:** A drawing layer draws shapes to a canvas and optionally includes rotate, scale, and alpha animations.
2. **Animation path:** An animation path layer describes a motion path for a drawing layer to follow. Paths in this type of layer are not visible in the exported file.

By default, layers are assumed to be drawing layers. To define a layer as an animation path, include the *type* property with a value of *animation* between the parentheses:

```
animationPath(type: animation);
```

| Property | Description               | Values                          |
|----------|---------------------------|---------------------------------|
| type/t   | Identifies the layer type | <b>drawing/d</b><br>animation/a |

## Animation

When any kind of animation is added to the document, the exported HTML includes a reference to an external JavaScript file called *Ai2CanvasAnimation.js*. If the file does not exist in the export directory, it is automatically created. If the file already exists, it is ignored. This means that custom JavaScript functions can safely be added to the file without fear of it being overwritten.

## Animation Clocks

All animation types (rotate, scale, alpha, and animation path) are driven by individual JavaScript *animation clocks*. Each of these clocks is synchronized with a master JavaScript timer function. Implementation details can be found in the *Ai2CanvasAnimation.js* file.

Each animation clock has 8 properties that behave similarly for each animation type.

| Property      | Description  | Values  |
|---------------|--|---|
| delay/del     | Length of initial delay in seconds (note that the delay <u>does not</u> repeat for each iteration) | <b>0.0</b><br>#   |
| direction/dir | Direction of the animation   | <b>none/n</b><br>(other values vary per animation type and are defined later) |

| Property            | Description  | Values                                  |
|---------------------|--|---|
| duration/dur        | Length of the animation in seconds   | <b>5.0</b><br>#                         |
| iterations/iter     | Number of times the animation repeats  | <b>infinite/i</b><br>#                  |
| multiplier/mult     | Multiplier to apply after the timing function                                  | <b>1.0</b><br>#                         |
| offset/off          | Offset to add after the multiplier (as a percentage of the clock's full range) | <b>0.0</b><br>#                         |
| reverses/rev        | Does the animation reverse direction?  | <b>no/n</b><br>yes/y                    |
| timing-function/t-f | Name of a JavaScript function that modifies time <i>t</i>                      | <b>linear/l</b><br><i>function name</i> |

## Drawing Layer Properties

The following properties are specific to drawing layers.

| Property               | Description   | Values  |
|------------------------|---|---|
| alpha-/a-              | Alpha prefix for the 8 animation clock properties described earlier. For example, <i>alpha-duration</i> sets the length of the alpha animation.   | (unique values below)   |
| alpha-direction/a-dir  | As-described in animation clock properties, with unique values  | <b>none/n</b><br>fade-in/f-i<br>fade-out/f-o  |
| animation/a            | Associates the drawing layer with an animation path layer   | <b>none/n</b><br><i>animation path layer name</i>   |
| crop/c                 | Crops the main canvas to the bounds of the artwork in the layer. Useful for positioning elements “off-screen.”  | <b>no/n</b><br>yes/y  |
| follow-orientation/f-o | Orientation (in degrees) of the “front” of the artwork to follow the associated animation path. 0 degrees faces directly to the right, and 90 degrees faces straight up.  | <b>none/n</b><br>#  |
| origin/o               | Modifies the position of the artwork origin. Specify x and y as percentages (e.g. 0.5 for 50%) of the width and height where the origin should be located. Note that values are allowed to position the origin outside the bounds of the artwork (e.g. -1.5, -1.0). | <b>normal/n</b><br><i>x, y</i><br>center/c<br>upper-left/ul<br>upper-right/ur<br>lower-right/lr<br>lower-left/l |

| Property               | Description   | Values  |
|------------------------|---|---|
| rasterize/rast         | Rasterizes the artwork in the layer to the specified file name. If the file exists, it is overwritten. Sometimes useful as a performance optimization. When combined with scale animation, remember that scaling up will result in a loss of quality. | <b>no/n</b><br><i>file name (without extension)</i>   |
| rotate-/r-             | Rotate prefix for the 8 animation clock properties described earlier. For example, <i>rotate-duration</i> sets the length of the rotate animation.  | (unique values below)                                 |
| rotate-direction/r-dir | As-described in animation clock properties, with unique values  | <b>none/n</b><br>clockwise/cw<br>counterclockwise/ccw |
| scale-/s-              | Scale prefix for the 8 animation clock properties described earlier. For example, <i>scale-duration</i> sets the length of the scale animation.   | (unique values below)                                 |
| scale-direction/s-dir  | As-described in animation clock properties, with unique values  | <b>none/n</b><br>grow/g<br>shrink/s                   |

## Animation Path Layer Properties

The following properties are specific to animation path layers.

| Property      | Description   | Values                                   |
|---------------|---|--|
|               | All animation clock properties are supported without a prefix. For example, <i>duration</i> sets the length of the animation. | (unique values below)                    |
| direction/dir | As-described in animation clock properties, with unique values  | <b>none/n</b><br>forward/f<br>backward/b |

## Timing Functions

A timing function accepts the current time from an animation clock (ranging from 0.0 – 1.0) and returns a new time value to be used in its place. Timing functions enable animations to progress in a more natural way.

Ai->Canvas includes 24 of [Robert Penner's easing equations](#) along with additional timing functions for both stepped and random changes. It is also easy to add custom timing functions.

These timing functions are included in the *Ai2CanvasAnimation.js* file.

|              |               |                 |
|--------------|---------------|-----------------|
| linear       |               |                 |
| random       | randomLimit   |                 |
| clockTick    |               |                 |
| zeroStep     | halfStep      | oneStep         |
| quadEaseIn   | quadEaseOut   | quadEaseInOut   |
| cubicEaseIn  | cubicEaseOut  | cubicEaseInOut  |
| quartEaseIn  | quartEaseOut  | quartEaseInOut  |
| quintEaseIn  | quintEaseOut  | quintEaseInOut  |
| sineEaseIn   | sineEaseOut   | sineEaseInOut   |
| expoEaseIn   | expoEaseOut   | expoEaseInOut   |
| circEaseIn   | circEaseOut   | circEaseInOut   |
| bounceEaseIn | bounceEaseOut | bounceEaseInOut |

## Triggers

Triggers are used to connect an *event* from one animation clock to a specific *action* on another. For example, a trigger can be used to start an animation when another animation completes.

Trigger = Event + Action

## Events

Each animation clock has four *events* that are referenced with their clock name prefix. For example, the *started* event of a rotate animation clock on a drawing layer called *draw* would be: *draw-rotate-started*

| Event    | Description  |
|----------|--|
| finished | The clock has run its full course and all iterations have completed.   |
| iterated | The animation clock has reached an iteration. This event is first fired after the first iteration, not when the clock initially starts.                          |
| started  | The animation clock has started. Note that this event can fire in the middle of an animation clock too (in the case that the clock was stopped for some reason). |
| stopped  | The animation clock has stopped. Like the <i>started</i> event, this could happen in the middle of an animation.   |

## Actions

When a trigger's event occurs, it causes an *action*. The trigger is attached to the clock that is to be controlled (not the clock that raises the event). There are no default triggers; they must be defined.

| Action       | Description  |
|--------------|--|
| fast-forward | "Fast forwards" the animation clock to beginning of its next iteration.          |
| reset        | Resets the animation clock to its initial conditions (but does not re-start it). |
| restart      | Resets the animation clock to its initial conditions and starts it.              |

| Action  | Description   |
|---------|---|
| reverse | Reverses the direction of the animation clock. Note that if the animation is in the middle of an iteration, the reversed motion is considered part of the same iteration (in other words, the iteration count does not change). |
| rewind  | “Rewinds” the animation clock to the beginning of its current iteration.  |
| start   | Starts (or restarts) the animation clock. If a <i>start</i> trigger is <u>not</u> defined, the animation clock is started when the HTML page is loaded.   |
| stop    | Stops/pauses the animation clock.   |
| toggle  | Starts the animation clock if it is stopped, and stops it if it is started.   |

## Example Triggers

To define a trigger for an animation clock, add a property to the layer name. The property name is the action on the animation clock, and the value is the layer, the animation clock, and the event that should cause the action (separated by dashes).

| Example Property: Value                | Description  |
|--|--|
| rotate-start: path1-stop               | Starts the rotate animation clock when the path1 animation clock stops.              |
| scale-rewind: redArrow-rotate-iterated | Rewinds the scale animation clock when the redArrow rotate animation clock iterates. |

## Other Uses for Events

Animation clock events can also be used to trigger custom, non-animation-clock features. For example, a sound could be played when an animation clock stops.

To respond to an animation clock event, first create a JavaScript function to handle the event. For example:

```
function playSound() {
    sound.play();
}
```

Then, at an appropriate place in JavaScript code (probably an “initialize”-style function), subscribe the custom function to the animation clock event:

```
drawDial.rotateClock.stopped.subscribe(playSound);
```

When the rotate animation clock on the custom *drawDial* drawing layer stops (e.g. raises the *stopped* event), the *playSound* function is called.