[ScrollMagic](http://docs.google.com/index.html)

* [Classes](http://docs.google.com/classes.list.html)
  + [Controller](http://docs.google.com/ScrollMagic.Controller.html)
  + [Scene](http://docs.google.com/ScrollMagic.Scene.html)
* [Events](http://docs.google.com/events.list.html)
  + [add](http://docs.google.com/ScrollMagic.Scene.html#event:add)
  + [change](http://docs.google.com/ScrollMagic.Scene.html#event:change)
  + [destroy](http://docs.google.com/ScrollMagic.Scene.html#event:destroy)
  + [end](http://docs.google.com/ScrollMagic.Scene.html#event:end)
  + [enter](http://docs.google.com/ScrollMagic.Scene.html#event:enter)
  + [leave](http://docs.google.com/ScrollMagic.Scene.html#event:leave)
  + [progress](http://docs.google.com/ScrollMagic.Scene.html#event:progress)
  + [remove](http://docs.google.com/ScrollMagic.Scene.html#event:remove)
  + [shift](http://docs.google.com/ScrollMagic.Scene.html#event:shift)
  + [start](http://docs.google.com/ScrollMagic.Scene.html#event:start)
  + [update](http://docs.google.com/ScrollMagic.Scene.html#event:update)
* [Plugins](http://docs.google.com/mixins.list.html)
  + [GSAP](http://docs.google.com/animation.GSAP.html)
  + [Velocity](http://docs.google.com/animation.Velocity.html)
  + [addIndicators](http://docs.google.com/debug.addIndicators.html)
  + [jQuery](http://docs.google.com/framework.jQuery.html)

Source: plugins/animation.gsap.js

/\*!  
 \* @file ScrollMagic GSAP Animation Plugin.  
 \*  
 \* requires: GSAP ~1.14  
 \* Powered by the Greensock Animation Platform (GSAP): http://www.greensock.com/js  
 \* Greensock License info at http://www.greensock.com/licensing/  
 \*/  
/\*\*  
 \* This plugin is meant to be used in conjunction with the Greensock Animation Plattform.   
 \* It offers an easy API to trigger Tweens or synchronize them to the scrollbar movement.  
 \*  
 \* Both the `lite` and the `max` versions of the GSAP library are supported.   
 \* The most basic requirement is `TweenLite`.  
 \*   
 \* To have access to this extension, please include `plugins/animation.gsap.js`.  
 \* @requires {@link http://greensock.com/gsap|GSAP ~1.14.x}  
 \* @mixin animation.GSAP  
 \*/  
(function (root, factory) {  
 if (typeof define === 'function' && define.amd) {  
 // AMD. Register as an anonymous module.  
 define(['ScrollMagic', 'TweenMax', 'TimelineMax'], factory);  
 } else if (typeof exports === 'object') {  
 // CommonJS  
 // Loads whole gsap package onto global scope.  
 require('gsap');  
 factory(require('scrollmagic'), TweenMax, TimelineMax);  
 } else {  
 // Browser globals  
 factory(root.ScrollMagic || (root.jQuery && root.jQuery.ScrollMagic), root.TweenMax || root.TweenLite, root.TimelineMax || root.TimelineLite);  
 }  
}(this, function(ScrollMagic, Tween, Timeline) {  
 "use strict";  
 var NAMESPACE = "animation.gsap";  
  
 // (BUILD) - REMOVE IN MINIFY - START  
 var  
 console = window.console || {},  
 err = Function.prototype.bind.call(console.error || console.log || function() {}, console);  
 if (!ScrollMagic) {  
 err("(" + NAMESPACE + ") -> ERROR: The ScrollMagic main module could not be found. Please make sure it's loaded before this plugin or use an asynchronous loader like requirejs.");  
 }  
 if (!Tween) {  
 err("(" + NAMESPACE + ") -> ERROR: TweenLite or TweenMax could not be found. Please make sure GSAP is loaded before ScrollMagic or use an asynchronous loader like requirejs.");  
 }  
 // (BUILD) - REMOVE IN MINIFY - END  
   
 /\*  
 \* ----------------------------------------------------------------  
 \* Extensions for Scene  
 \* ----------------------------------------------------------------  
 \*/  
 /\*\*  
 \* Every instance of ScrollMagic.Scene now accepts an additional option.   
 \* See {@link ScrollMagic.Scene} for a complete list of the standard options.  
 \* @memberof! animation.GSAP#  
 \* @method new ScrollMagic.Scene(options)  
 \* @example  
 \* var scene = new ScrollMagic.Scene({tweenChanges: true});  
 \*  
 \* @param {object} [options] - Options for the Scene. The options can be updated at any time.  
 \* @param {boolean} [options.tweenChanges=false] - Tweens Animation to the progress target instead of setting it.   
 Does not affect animations where duration is `0`.  
 \*/  
 /\*\*  
 \* \*\*Get\*\* or \*\*Set\*\* the tweenChanges option value.   
 \* This only affects scenes with a duration. If `tweenChanges` is `true`, the progress update when scrolling will not be immediate, but instead the animation will smoothly animate to the target state.   
 \* For a better understanding, try enabling and disabling this option in the [Scene Manipulation Example](../examples/basic/scene\_manipulation.html).  
 \* @memberof! animation.GSAP#  
 \* @method Scene.tweenChanges  
 \*   
 \* @example  
 \* // get the current tweenChanges option  
 \* var tweenChanges = scene.tweenChanges();  
 \*  
 \* // set new tweenChanges option  
 \* scene.tweenChanges(true);  
 \*  
 \* @fires {@link Scene.change}, when used as setter  
 \* @param {boolean} [newTweenChanges] - The new tweenChanges setting of the scene.  
 \* @returns {boolean} `get` - Current tweenChanges option value.  
 \* @returns {Scene} `set` - Parent object for chaining.  
 \*/  
 // add option (TODO: DOC (private for dev))  
 ScrollMagic.Scene.addOption(  
 "tweenChanges", // name  
 false, // default  
 function (val) { // validation callback  
 return !!val;  
 }  
 );  
 // extend scene  
 ScrollMagic.Scene.extend(function () {  
 var Scene = this,  
 \_tween;  
  
 // (BUILD) - REMOVE IN MINIFY - START  
 var log = function () {  
 if (Scene.\_log) { // not available, when main source minified  
 Array.prototype.splice.call(arguments, 1, 0, "(" + NAMESPACE + ")", "->");  
 Scene.\_log.apply(this, arguments);  
 }  
 };  
 // (BUILD) - REMOVE IN MINIFY - END  
  
 // set listeners  
 Scene.on("progress.plugin\_gsap", function () {  
 updateTweenProgress();  
 });  
 Scene.on("destroy.plugin\_gsap", function (e) {  
 Scene.removeTween(e.reset);  
 });  
  
 /\*\*  
 \* Update the tween progress to current position.  
 \* @private  
 \*/  
 var updateTweenProgress = function () {  
 if (\_tween) {  
 var  
 progress = Scene.progress(),  
 state = Scene.state();  
 if (\_tween.repeat && \_tween.repeat() === -1) {  
 // infinite loop, so not in relation to progress  
 if (state === 'DURING' && \_tween.paused()) {  
 \_tween.play();  
 } else if (state !== 'DURING' && !\_tween.paused()) {  
 \_tween.pause();  
 }  
 } else if (progress != \_tween.progress()) { // do we even need to update the progress?  
 // no infinite loop - so should we just play or go to a specific point in time?  
 if (Scene.duration() === 0) {  
 // play the animation  
 if (progress > 0) { // play from 0 to 1  
 \_tween.play();  
 } else { // play from 1 to 0  
 \_tween.reverse();  
 }  
 } else {  
 // go to a specific point in time  
 if (Scene.tweenChanges() && \_tween.tweenTo) {  
 // go smooth  
 \_tween.tweenTo(progress \* \_tween.duration());  
 } else {  
 // just hard set it  
 \_tween.progress(progress).pause();  
 }  
 }  
 }  
 }  
 };  
  
 /\*\*  
 \* Add a tween to the scene.   
 \* If you want to add multiple tweens, add them into a GSAP Timeline object and supply it instead (see example below).   
 \*   
 \* If the scene has a duration, the tween's duration will be projected to the scroll distance of the scene, meaning its progress will be synced to scrollbar movement.   
 \* For a scene with a duration of `0`, the tween will be triggered when scrolling forward past the scene's trigger position and reversed, when scrolling back.   
 \* To gain better understanding, check out the [Simple Tweening example](../examples/basic/simple\_tweening.html).  
 \*  
 \* Instead of supplying a tween this method can also be used as a shorthand for `TweenMax.to()` (see example below).  
 \* @memberof! animation.GSAP#  
 \*  
 \* @example  
 \* // add a single tween directly  
 \* scene.setTween(TweenMax.to("obj"), 1, {x: 100});  
 \*  
 \* // add a single tween via variable  
 \* var tween = TweenMax.to("obj"), 1, {x: 100};  
 \* scene.setTween(tween);  
 \*  
 \* // add multiple tweens, wrapped in a timeline.  
 \* var timeline = new TimelineMax();  
 \* var tween1 = TweenMax.from("obj1", 1, {x: 100});  
 \* var tween2 = TweenMax.to("obj2", 1, {y: 100});  
 \* timeline  
 \* .add(tween1)  
 \* .add(tween2);  
 \* scene.addTween(timeline);  
 \*  
 \* // short hand to add a TweenMax.to() tween  
 \* scene.setTween("obj3", 0.5, {y: 100});  
 \*  
 \* // short hand to add a TweenMax.to() tween for 1 second  
 \* // this is useful, when the scene has a duration and the tween duration isn't important anyway  
 \* scene.setTween("obj3", {y: 100});  
 \*  
 \* @param {(object|string)} TweenObject - A TweenMax, TweenLite, TimelineMax or TimelineLite object that should be animated in the scene. Can also be a Dom Element or Selector, when using direct tween definition (see examples).  
 \* @param {(number|object)} duration - A duration for the tween, or tween parameters. If an object containing parameters are supplied, a default duration of 1 will be used.  
 \* @param {object} params - The parameters for the tween  
 \* @returns {Scene} Parent object for chaining.  
 \*/  
 Scene.setTween = function (TweenObject, duration, params) {  
 var newTween;  
 if (arguments.length > 1) {  
 if ( arguments.length < 3) {  
 params = duration;  
 duration = 1;  
 }  
 TweenObject = Tween.to(TweenObject, duration, params);  
 }  
 try {  
 // wrap Tween into a Timeline Object if available to include delay and repeats in the duration and standardize methods.  
 if (Timeline) {  
 newTween = new Timeline({smoothChildTiming: true})  
 .add(TweenObject);  
 } else {  
 newTween = TweenObject;  
 }  
 newTween.pause();  
 } catch (e) {  
 log(1, "ERROR calling method 'setTween()': Supplied argument is not a valid TweenObject");  
 return Scene;  
 }  
 if (\_tween) { // kill old tween?  
 Scene.removeTween();  
 }  
 \_tween = newTween;  
  
 // some properties need to be transferred it to the wrapper, otherwise they would get lost.  
 if (TweenObject.repeat && TweenObject.repeat() === -1) {// TweenMax or TimelineMax Object?  
 \_tween.repeat(-1);  
 \_tween.yoyo(TweenObject.yoyo());  
 }  
 // (BUILD) - REMOVE IN MINIFY - START  
 // Some tween validations and debugging helpers  
  
 if (Scene.tweenChanges() && !\_tween.tweenTo) {  
 log(2, "WARNING: tweenChanges will only work if the TimelineMax object is available for ScrollMagic.");  
 }  
  
 // check if there are position tweens defined for the trigger and warn about it :)  
 if (\_tween && Scene.controller() && Scene.triggerElement() && Scene.loglevel() >= 2) {// controller is needed to know scroll direction.  
 var  
 triggerTweens = Tween.getTweensOf(Scene.triggerElement()),  
 vertical = Scene.controller().info("vertical");  
 triggerTweens.forEach(function (value, index) {  
 var  
 tweenvars = value.vars.css || value.vars,  
 condition = vertical ? (tweenvars.top !== undefined || tweenvars.bottom !== undefined) : (tweenvars.left !== undefined || tweenvars.right !== undefined);  
 if (condition) {  
 log(2, "WARNING: Tweening the position of the trigger element affects the scene timing and should be avoided!");  
 return false;  
 }  
 });  
 }  
  
 // warn about tween overwrites, when an element is tweened multiple times  
 if (parseFloat(TweenLite.version) >= 1.14) { // onOverwrite only present since GSAP v1.14.0  
 var  
 list = \_tween.getChildren ? \_tween.getChildren(true, true, false) : [\_tween], // get all nested tween objects  
 newCallback = function () {  
 log(2, "WARNING: tween was overwritten by another. To learn how to avoid this issue see here: https://github.com/janpaepke/ScrollMagic/wiki/WARNING:-tween-was-overwritten-by-another");  
 };  
 for (var i=0, thisTween, oldCallback; i<list.length; i++) {  
 /\*jshint loopfunc: true \*/  
 thisTween = list[i];  
 if (oldCallback !== newCallback) { // if tweens is added more than once  
 oldCallback = thisTween.vars.onOverwrite;  
 thisTween.vars.onOverwrite = function () {  
 if (oldCallback) {  
 oldCallback.apply(this, arguments);  
 }  
 newCallback.apply(this, arguments);  
 };  
 }  
 }  
 }  
 // (BUILD) - REMOVE IN MINIFY - END  
 log(3, "added tween");  
  
 updateTweenProgress();  
 return Scene;  
 };  
  
 /\*\*  
 \* Remove the tween from the scene.   
 \* This will terminate the control of the Scene over the tween.  
 \*  
 \* Using the reset option you can decide if the tween should remain in the current state or be rewound to set the target elements back to the state they were in before the tween was added to the scene.  
 \* @memberof! animation.GSAP#  
 \*  
 \* @example  
 \* // remove the tween from the scene without resetting it  
 \* scene.removeTween();  
 \*  
 \* // remove the tween from the scene and reset it to initial position  
 \* scene.removeTween(true);  
 \*  
 \* @param {boolean} [reset=false] - If `true` the tween will be reset to its initial values.  
 \* @returns {Scene} Parent object for chaining.  
 \*/  
 Scene.removeTween = function (reset) {  
 if (\_tween) {  
 if (reset) {  
 \_tween.progress(0).pause();  
 }  
 \_tween.kill();  
 \_tween = undefined;  
 log(3, "removed tween (reset: " + (reset ? "true" : "false") + ")");  
 }  
 return Scene;  
 };  
  
 });  
}));

© Jan Paepke 2015

Documentation generated by [JSDoc 3.5.5](https://github.com/jsdoc3/jsdoc) using a customized version of the [DocStrap template](https://github.com/terryweiss/docstrap).