

# Director Class

Extends [EventTarget](#)

Module: [cc](#)

ATTENTION: USE cc.director INSTEAD OF cc.Director.

cc.director is a singleton object which manage your game's logic flow.

Since the cc.director is a singleton, you don't need to call any constructor or create functions,

the standard way to use it is by calling:

- cc.director.methodName();

It creates and handle the main Window and manages how and when to execute the Scenes.

The cc.director is also responsible for:

- initializing the OpenGL context
- setting the OpenGL pixel format (default on is RGB565)
- setting the OpenGL buffer depth (default on is 0-bit)
- setting the color for clear screen (default one is BLACK)
- setting the projection (default one is 3D)
- setting the orientation (default one is Portrait)

The cc.director also sets the default OpenGL context:

- GL\_TEXTURE\_2D is enabled
- GL\_VERTEX\_ARRAY is enabled
- GL\_COLOR\_ARRAY is enabled
- GL\_TEXTURE\_COORD\_ARRAY is enabled

cc.director also synchronizes timers with the refresh rate of the display.

Features and Limitations:

- Scheduled timers & drawing are synchronizes with the refresh rate of the display
- Only supports animation intervals of 1/60 1/30 & 1/15

## Index

### Properties

- [EVENT\\_PROJECTION\\_CHANGED](#) String The event projection changed of cc.Director.
- [EVENT\\_BEFORE\\_SCENE\\_LOADING](#) String The event which will be triggered before loading a new scene.
- [EVENT\\_BEFORE\\_SCENE\\_LAUNCH](#) String The event which will be triggered before launching a new scene.
- [EVENT\\_AFTER\\_SCENE\\_LAUNCH](#) String The event which will be triggered after launching a new scene.
- [EVENT\\_BEFORE\\_UPDATE](#) String The event which will be triggered at the beginning of every frame.

- `EVENT_AFTER_UPDATE` String The event which will be triggered after engine and components update logic.
- `EVENT_BEFORE_VISIT` String The event is deprecated since v2.0, please use `cc.Director.EVENT_BEFORE_DRAW` instead
- `EVENT_AFTER_VISIT` String The event is deprecated since v2.0, please use `cc.Director.EVENT_BEFORE_DRAW` instead
- `EVENT_BEFORE_DRAW` String The event which will be triggered before the rendering process.
- `EVENT_AFTER_DRAW` String The event which will be triggered after the rendering process.
- `PROJECTION_2D` Number Constant for 2D projection (orthogonal projection)
- `PROJECTION_3D` Number Constant for 3D projection with a `fovy=60`, `znear=0.5f` and `zfar=1500`.
- `PROJECTION_CUSTOM` Number Constant for custom projection, if `cc.Director`'s projection set to it, it calls "updateProjection" on the projection delegate.
- `PROJECTION_DEFAULT` Number Constant for default projection of `cc.Director`, default projection is 2D projection

## Methods

- `convertToGL` Converts a view coordinate to an WebGL coordinate...
- `convertToUI` Converts an OpenGL coordinate to a view coordinate...
- `end` End the life of director in the next frame
- `getWinSize` Returns the size of the WebGL view in points....
- `getWinSizeInPixels` On Mac `winSize` and `winSizeInPixels` return the same value.
- `pause` Pause the director's ticker, only involve the game logic execution.
- `runSceneImmediate` Run a scene.
- `runScene` Run a scene.
- `loadScene` Loads the scene by its name.
- `preloadScene` Preloads the scene to reduces loading time.
- `_loadSceneByUuid` Loads the scene by its uuid.
- `resume` Resume game logic execution after pause, if the current scene is not paused, nothing will happen.
- `setDepthTest` Enables or disables WebGL depth test....
- `setClearColor` Set color for clear screen....
- `getRunningScene` Returns current logic Scene.
- `getScene` Returns current logic Scene.
- `getAnimationInterval` Returns the FPS value.
- `setAnimationInterval` Sets animation interval, this doesn't control the main loop.
- `getDeltaTime` Returns the delta time since last frame.
- `getTotalFrames` Returns how many frames were called since the director started.
- `isPaused` Returns whether or not the Director is paused.
- `getScheduler` Returns the `cc.Scheduler` associated with this director.
- `setScheduler` Sets the `cc.Scheduler` associated with this director.
- `getActionManager` Returns the `cc.ActionManager` associated with this director.
- `setActionManager` Sets the `cc.ActionManager` associated with this director.
- `getCollisionManager` Returns the `cc.CollisionManager` associated with this director.
- `getPhysicsManager` Returns the `cc.PhysicsManager` associated with this director.
- `hasEventListener` Checks whether the `EventTarget` object has any callback registered for a specific type of event.
- `on` Register an callback of a specific event type on the `EventTarget`.
- `off` Removes the listeners previously registered with the same type, callback, target and or `useCapture`,...

- `targetOff` Removes all callbacks previously registered with the same target (passed as parameter).
- `once` Register an callback of a specific event type on the EventTarget,...
- `emit` Trigger an event directly with the event name and necessary arguments.
- `dispatchEvent` Send an event with the event object.

## Events

- `cc.Director.EVENT_BEFORE_SCENE_LOADING` The event which will be triggered before loading a new scene.
- `cc.Director.EVENT_AFTER_SCENE_LAUNCH` The event which will be triggered after launching a new scene.
- `cc.Director.EVENT_BEFORE_UPDATE` The event which will be triggered at the beginning of every frame.
- `cc.Director.EVENT_AFTER_UPDATE` The event which will be triggered after engine and components update logic.
- `cc.Director.EVENT_BEFORE_DRAW` The event which will be triggered before the rendering process.
- `cc.Director.EVENT_AFTER_DRAW` The event which will be triggered after the rendering process.

## Details

### *Properties*

#### EVENT\_PROJECTION\_CHANGED

The event projection changed of cc.Director. This event will not get triggered since v2.0

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/CCDirector.js:980</a>
Deprecated	since v2.0

#### EVENT\_BEFORE\_SCENE\_LOADING

The event which will be triggered before loading a new scene.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/CCDirector.js:996</a>

## EVENT\_BEFORE\_SCENE\_LAUNCH

The event which will be triggered before launching a new scene.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/CCDirector.js:1011</a>

## EVENT\_AFTER\_SCENE\_LAUNCH

The event which will be triggered after launching a new scene.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/CCDirector.js:1026</a>

## EVENT\_BEFORE\_UPDATE

The event which will be triggered at the beginning of every frame.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/CCDirector.js:1040</a>

## EVENT\_AFTER\_UPDATE

The event which will be triggered after engine and components update logic.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/CCDirector.js:1054</a>

## EVENT\_BEFORE\_VISIT

The event is deprecated since v2.0, please use cc.Director.EVENT\_BEFORE\_DRAW instead

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/CCDirector.js:1063</a>
Deprecated	since v2.0

## EVENT\_AFTER\_VISIT

The event is deprecated since v2.0, please use cc.Director.EVENT\_BEFORE\_DRAW instead

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/CCDirector.js:1073</a>
Deprecated	since v2.0

## EVENT\_BEFORE\_DRAW

The event which will be triggered before the rendering process.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/CCDirector.js:1088</a>

## EVENT\_AFTER\_DRAW

The event which will be triggered after the rendering process.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/CCDirector.js:1102</a>

## PROJECTION\_2D

Constant for 2D projection (orthogonal projection)

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/CCDirector.js:1113</a>
Deprecated	since v2.0

## PROJECTION\_3D

Constant for 3D projection with a fovy=60, znear=0.5f and zfar=1500.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/CCDirector.js:1123</a>

meta	description
Deprecated	since v2.0

## PROJECTION\_CUSTOM

Constant for custom projection, if cc.Director's projection set to it, it calls "updateProjection" on the projection delegate.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/CCDirector.js:1133</a>
Deprecated	since v2.0

## PROJECTION\_DEFAULT

Constant for default projection of cc.Director, default projection is 2D projection

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/CCDirector.js:1143</a>
Deprecated	since v2.0

## *Methods*

### convertToGL

Converts a view coordinate to an WebGL coordinate  
 Useful to convert (multi) touches coordinates to the current layout (portrait or landscape)  
 Implementation can be found in CCDirectorWebGL.

meta	description
Returns	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/CCDirector.js:230</a>
Deprecated	since v2.0

## Parameters

- uiPoint [Vec2](#)

### convertToUI

Converts an OpenGL coordinate to a view coordinate

Useful to convert node points to window points for calls such as glScissor  
Implementation can be found in CCDirectorWebGL.

meta	description
Returns	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/CCDirector.js:252</a>
Deprecated	since v2.0

## Parameters

- glPoint [Vec2](#)

### end

End the life of director in the next frame

meta	description
Defined in	<a href="#">cocos2d/core/CCDirector.js:281</a>

## getWinSize

Returns the size of the WebGL view in points.

It takes into account any possible rotation (device orientation) of the window.

meta	description
Returns	<a href="#">Size</a>
Defined in	<a href="#">cocos2d/core/CCDirector.js:289</a>
Deprecated	since v2.0

## getWinSizeInPixels

Returns the size of the OpenGL view in pixels.

It takes into account any possible rotation (device orientation) of the window.

On Mac winSize and winSizeInPixels return the same value. (The pixel here refers to the resource resolution. If you want to get the physics resolution of device, you need to use cc.view.getFrameSize())

meta	description
Returns	<a href="#">Size</a>
Defined in	<a href="#">cocos2d/core/CCDirector.js:302</a>
Deprecated	since v2.0

## pause

Pause the director's ticker, only involve the game logic execution. It won't pause the rendering process nor the event manager. If you want to pause the entire game including rendering, audio and event, please use Game.pause

meta	description
Defined in	<a href="#">cocos2d/core/CCDirector.js:319</a>

## runSceneImmediate

Run a scene. Replaces the running scene with a new one or enter the first scene. The new scene will be launched immediately.

meta	description
Defined in	<a href="#">cocos2d/core/CCDirector.js:403</a>

### Parameters

- `scene Scene` The need run scene.
- `onBeforeLoadScene Function` The function invoked at the scene before loading.
- `onLaunched Function` The function invoked at the scene after launch.

## runScene

Run a scene. Replaces the running scene with a new one or enter the first scene. The new scene will be launched at the end of the current frame.

meta	description
Defined in	<a href="#">cocos2d/core/CCDirector.js:482</a>

### Parameters

- `scene Scene` The need run scene.
- `onBeforeLoadScene Function` The function invoked at the scene before loading.
- `onLaunched Function` The function invoked at the scene after launch.

## loadScene

Loads the scene by its name.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/CCDirector.js:539</a>

## Parameters

- `sceneName` [String](#) The name of the scene to load.
- `onLaunched` [Function](#) callback, will be called after scene launched.

## preloadScene

Preloads the scene to reduces loading time. You can call this method at any time you want. After calling this method, you still need to launch the scene by `cc.director.loadScene`. It will be totally fine to call `cc.director.loadScene` at any time even if the preloading is not yet finished, the scene will be launched after loaded automatically.

meta	description
Defined in	<a href="#">cocos2d/core/CCDirector.js:567</a>

## Parameters

- `sceneName` [String](#) The name of the scene to preload.
- `onProgress` [Function](#) callback, will be called when the load progression change.
- `completedCount` [Number](#) The number of the items that are already completed
- `totalCount` [Number](#) The total number of the items
- `item` [Object](#) The latest item which flow out the pipeline
- `onLoaded` [Function](#) callback, will be called after scene loaded.
- `error` [Error](#) null or the error object.

## \_loadSceneByUuid

Loads the scene by its uuid.

meta	description
Defined in	<a href="#">cocos2d/core/CCDirector.js:613</a>

## Parameters

- `uuid` [String](#) the uuid of the scene asset to load
- `onLaunched` [Function](#)
- `onUnloaded` [Function](#)
- `dontRunScene` [Boolean](#) Just download and initialize the scene but will not launch it, only take effect in the Editor.

## resume

Resume game logic execution after pause, if the current scene is not paused, nothing will happen.

meta	description
Defined in	<a href="#">cocos2d/core/CCDirector.js:676</a>

## setDepthTest

Enables or disables WebGL depth test.

Implementation can be found in [CCDirectorCanvas.js](#)/[CCDirectorWebGL.js](#)

meta	description
Defined in	<a href="#">cocos2d/core/CCDirector.js:695</a>
Deprecated	since v2.0

## Parameters

- `on Boolean`

## setClearColor

Set color for clear screen.

(Implementation can be found in [CCDirectorCanvas.js](#)/[CCDirectorWebGL.js](#))

meta	description
Defined in	<a href="#">cocos2d/core/CCDirector.js:711</a>
Deprecated	since v2.0

## Parameters

- `clearColor Color`

## getRunningScene

Returns current logic Scene.

meta	description
Returns	<a href="#">Scene</a>
Defined in	<a href="#">cocos2d/core/CCDirector.js:729</a>
Deprecated	since v2.0

## getScene

Returns current logic Scene.

meta	description
Returns	<a href="#">Scene</a>
Defined in	<a href="#">cocos2d/core/CCDirector.js:741</a>

## Examples

```
// This will help you to get the Canvas node in scene
cc.director.getScene().getChildByName('Canvas');
```

## getAnimationInterval

Returns the FPS value. Please use Game.setFrameRate to control animation interval.

meta	description
Returns	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/CCDirector.js:754</a>
Deprecated	since v2.0

## setAnimationInterval

Sets animation interval, this doesn't control the main loop. To control the game's frame rate overall, please use Game.setFrameRate

meta	description
Defined in	<a href="#">cocos2d/core/CCDirector.js:765</a>
Deprecated	since v2.0

## Parameters

- **value** [Number](#) The animation interval desired.

## getDeltaTime

Returns the delta time since last frame.

meta	description
Returns	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/CCDirector.js:776</a>

## getTotalFrames

Returns how many frames were called since the director started.

meta	description
Returns	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/CCDirector.js:786</a>

## isPaused

Returns whether or not the Director is paused.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/CCDirector.js:796</a>

## getScheduler

Returns the cc.Scheduler associated with this director.

meta	description
Returns	<a href="#">Scheduler</a>
Defined in	<a href="#">cocos2d/core/CCDirector.js:806</a>

## setScheduler

Sets the cc.Scheduler associated with this director.

meta	description
Defined in	<a href="#">cocos2d/core/CCDirector.js:816</a>

## Parameters

- `scheduler` [Scheduler](#)

## getActionManager

Returns the cc.ActionManager associated with this director.

meta	description
Returns	<a href="#">ActionManager</a>
Defined in	<a href="#">cocos2d/core/CCDirector.js:828</a>

## setActionManager

Sets the cc.ActionManager associated with this director.

meta	description
Defined in	<a href="#">cocos2d/core/CCDirector.js:837</a>

### Parameters

- `actionManager` [ActionManager](#)

## getCollisionManager

Returns the cc.CollisionManager associated with this director.

meta	description
Returns	<a href="#">CollisionManager</a>
Defined in	<a href="#">cocos2d/core/CCDirector.js:863</a>

## getPhysicsManager

Returns the cc.PhysicsManager associated with this director.

meta	description
Returns	<a href="#">PhysicsManager</a>
Defined in	<a href="#">cocos2d/core/CCDirector.js:873</a>

## hasEventListener

Checks whether the EventTarget object has any callback registered for a specific type of event.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/event/event-target.js:68</a>

## Parameters

- type [String](#) The type of event.

on

Register an callback of a specific event type on the EventTarget. This type of event should be triggered via `emit`.

meta	description
Returns	<a href="#">Function</a>
Defined in	<a href="#">cocos2d/core/event/event-target.js:76</a>

## Parameters

- type [String](#) A string representing the event type to listen for.
- callback [Function](#) The callback that will be invoked when the event is dispatched.
- The callback is ignored if it is a duplicate (the callbacks are unique).
- arg1 Any arg1
- arg2 Any arg2
- arg3 Any arg3
- arg4 Any arg4
- arg5 Any arg5
- target [Object](#) The target (this object) to invoke the callback, can be null

## Examples

```
eventTarget.on('fire', function (event) {
    cc.log("fire in the hole");
}, node);
```

off

Removes the listeners previously registered with the same type, callback, target and or useCapture, if only type is passed as parameter, all listeners registered with that type will be removed.

meta	description
Defined in	<a href="#">cocos2d/core/event/event-target.js:117</a>

## Parameters

- type [String](#) A string representing the event type being removed.
- callback [Function](#) The callback to remove.
- target [Object](#) The target (this object) to invoke the callback, if it's not given, only callback without target will be removed

## Examples

```
// register fire eventListener
var callback = eventTarget.on('fire', function (event) {
    cc.log("fire in the hole");
}, target);
// remove fire event listener
eventTarget.off('fire', callback, target);
// remove all fire event listeners
eventTarget.off('fire');
```

## targetOff

Removes all callbacks previously registered with the same target (passed as parameter). This is not for removing all listeners in the current event target, and this is not for removing all listeners the target parameter have registered. It's only for removing all listeners (callback and target couple) registered on the current event target by the target parameter.

meta	description
Defined in	<a href="#">cocos2d/core/event/event-target.js:151</a>

## Parameters

- target [Object](#) The target to be searched for all related listeners

## once

Register an callback of a specific event type on the EventTarget, the callback will remove itself after the first time it is triggered.

meta	description
Defined in	<a href="#">cocos2d/core/event/event-target.js:164</a>

## Parameters

- type [String](#) A string representing the event type to listen for.
- callback [Function](#) The callback that will be invoked when the event is dispatched.
- The callback is ignored if it is a duplicate (the callbacks are unique).
- arg1 Any arg1
- arg2 Any arg2
- arg3 Any arg3
- arg4 Any arg4
- arg5 Any arg5
- target [Object](#) The target (this object) to invoke the callback, can be null

## Examples

```
eventTarget.once('fire', function (event) {
    cc.log("this is the callback and will be invoked only once");
}, node);
```

## emit

Trigger an event directly with the event name and necessary arguments.

meta	description
Defined in	<a href="#">cocos2d/core/event/event-target.js:201</a>

## Parameters

- type [String](#) event type
- arg1 Any First argument
- arg2 Any Second argument
- arg3 Any Third argument
- arg4 Any Fourth argument
- arg5 Any Fifth argument

## Examples

```
eventTarget.emit('fire', event);
eventTarget.emit('fire', message, emitter);
```

## dispatchEvent

Send an event with the event object.

meta	description
Defined in	<a href="#">cocos2d/core/event/event-target.js:221</a>

### Parameters

- `event` [Event](#)

### *Events*

#### **cc.Director.EVENT\_BEFORE\_SCENE\_LOADING Event**

Module: [cc](#)

The event which will be triggered before loading a new scene.

## Index

### Details

#### **cc.Director.EVENT\_AFTER\_SCENE\_LAUNCH Event**

Module: [cc](#)

The event which will be triggered after launching a new scene.

## Index

### Details

#### **cc.Director.EVENT\_BEFORE\_UPDATE Event**

Module: [cc](#)

The event which will be triggered at the beginning of every frame.

## Index

### Details

## **cc.Director.EVENT\_AFTER\_UPDATE Event**

Module: [cc](#)

The event which will be triggered after engine and components update logic.

## **Index**

### **Details**

## **cc.Director.EVENT\_BEFORE\_DRAW Event**

Module: [cc](#)

The event which will be triggered before the rendering process.

## **Index**

### **Details**

## **cc.Director.EVENT\_AFTER\_DRAW Event**

Module: [cc](#)

The event which will be triggered after the rendering process.

## **DistanceJoint Class**

Extends [Joint](#)

Module: [cc](#) Parent Module: [cc](#)

A distance joint constrains two points on two bodies to remain at a fixed distance from each other. You can view this as a massless, rigid rod.

## **Index**

### Properties

- `distance` Number The distance separating the two ends of the joint.
- `frequency` Number The spring frequency.
- `0` Number The damping ratio.
- `anchor` Vec2 The anchor of the rigidbody.
- `connectedAnchor` Vec2 The anchor of the connected rigidbody.
- `connectedBody` RigidBody The rigidbody to which the other end of the joint is attached.
- `collideConnected` Boolean Should the two rigid bodies connected with this joint collide with each other?
- `__eventTargets` Array Register all related EventTargets,...
- `node` Node The node this component is attached to.
- `uuid` String The uuid for editor.

- `_enabled Boolean`
- `enabled Boolean` indicates whether this component is enabled or not.
- `enabledInHierarchy Boolean` indicates whether this component is enabled and its node is also active in the hierarchy.
- `_isOnLoadCalled Number` Returns a value which used to indicate the onLoad get called or not.
- `_name String`
- `_objFlags Number`
- `name String` The name of the object.
- `isValid Boolean` Indicates whether the object is not yet destroyed.

## Methods

- `apply` Apply current changes to joint, this will regenerate inner box2d joint.
- `getWorldAnchor` Get the anchor point on rigidbody in world coordinates.
- `getWorldConnectedAnchor` Get the anchor point on connected rigidbody in world coordinates.
- `getReactionForce` Gets the reaction force of the joint.
- `getReactionTorque` Gets the reaction torque of the joint.
- `update` This is a lifecycle method.
- `lateUpdate` This is a lifecycle method.
- `__preload` \_\_preload is called before every onLoad.
- `onLoad` When attaching to an active node or its node first activated.
- `start` Called before all scripts' update if the Component is enabled the first time.
- `onEnable` This is a lifecycle method.
- `onDisable` This is a lifecycle method.
- `onDestroy` This is a lifecycle method.
- `onFocusInEditor`
- `onLostFocusInEditor`
- `resetInEditor` Called to initialize the component or node's properties when adding the component the first time or when the Reset command is used.
- `addComponent` Adds a component class to the node.
- `getComponent` Returns the component of supplied type if the node has one attached, null if it doesn't....
- `getComponents` Returns all components of supplied Type in the node.
- `getComponentInChildren` Returns the component of supplied type in any of its children using depth first search.
- `getComponentsInChildren` Returns the components of supplied type in self or any of its children using depth first search.
- `_getLocalBounds` If the component's bounding box is different from the node's, you can implement this method to supply
- `onRestore` for undo/redo operation.
- `schedule` Schedules a custom selector....
- `scheduleOnce` Schedules a callback function that runs only once, with a delay of 0 or larger.
- `unschedule` Unschedules a custom callback function.
- `unscheduleAllCallbacks` unschedule all scheduled callback functions: custom callback functions, and the 'update' callback function....
- `destroy` Actual object destruction will delayed until before rendering.
- `_destruct` Clear all references in the instance.
- `_onPreDestroy` Called before the object being destroyed.
- `_serialize` The customized serialization for this object.
- `_deserialize` Init this object from the custom serialized data.

## Details

### *Properties*

distance

The distance separating the two ends of the joint.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCDistanceJoint.js:56</a>

frequency

The spring frequency.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCDistanceJoint.js:77</a>

0

The damping ratio.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCDistanceJoint.js:98</a>

anchor

The anchor of the rigidbody.

meta	description
Type	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCJoint.js:49</a>

#### connectedAnchor

The anchor of the connected rigidbody.

meta	description
Type	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCJoint.js:61</a>

#### connectedBody

The rigidbody to which the other end of the joint is attached.

meta	description
Type	<a href="#">RigidBody</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCJoint.js:74</a>

#### collideConnected

Should the two rigid bodies connected with this joint collide with each other?

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCJoint.js:88</a>

## \_\_eventTargets

Register all related EventTargets, all event callbacks will be removed in \_onPreDestroy

meta	description
Type	<a href="#">Array</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:61</a>

## node

The node this component is attached to. A component is always attached to a node.

meta	description
Type	<a href="#">Node</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:75</a>

## Examples

```
cc.log(comp.node);
```

## uuid

The uuid for editor.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:106</a>

## Examples

```
cc.log(comp.uuid);
```

### \_enabled

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:147</a>

### **enabled**

indicates whether this component is enabled or not.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:154</a>

### Examples

```
comp.enabled = true;  
cc.log(comp.enabled);
```

### **enabledInHierarchy**

indicates whether this component is enabled and its node is also active in the hierarchy.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:185</a>

### Examples

```
cc.log(comp.enabledInHierarchy);
```

### \_isOnLoadCalled

Returns a value which used to indicate the onLoad get called or not.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:201</a>

## Examples

```
cc.log(this.\_isOnLoadCalled > 0);
```

`_name`

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:76</a>

`_objFlags`

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:83</a>

`name`

The name of the object.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:243</a>

## Examples

```
obj.name = "New Obj";
```

### isValid

Indicates whether the object is not yet destroyed. (It will not be available after being destroyed)  
When an object's `destroy` is called, it is actually destroyed after the end of this frame.  
So `isValid` will return false from the next frame, while `isValid` in the current frame will still be true.  
If you want to determine whether the current frame has called `destroy`, use `cc.isValid(obj, true)`, but this is often caused by a particular logical requirements, which is not normally required.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:261</a>

## Examples

```
var node = new cc.Node();
cc.log(node.isValid);      // true
node.destroy();
cc.log(node.isValid);      // true, still valid in this frame
// after a frame...
cc.log(node.isValid);      // false, destroyed in the end of last frame
```

## Methods

### apply

Apply current changes to joint, this will regenerate inner box2d joint.

meta	description
Defined in	<a href="#">cocos2d/core/physics/joint/CCJoint.js:115</a>

### getWorldAnchor

Get the anchor point on rigidbody in world coordinates.

meta	description
Returns	<a href="#">Vec2</a>

meta	description
Defined in	<a href="#">cocos2d/core/physics/joint/CCJoint.js:127</a>

### getWorldConnectedAnchor

Get the anchor point on connected rigidbody in world coordinates.

meta	description
Returns	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCJoint.js:143</a>

### getReactionForce

Gets the reaction force of the joint.

meta	description
Returns	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCJoint.js:159</a>

### Parameters

- `timeStep` [Number](#) The time to calculate the reaction force for.

### getReactionTorque

Gets the reaction torque of the joint.

meta	description
Returns	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCJoint.js:176</a>

## Parameters

- `timeStep` [Number](#) The time to calculate the reaction torque for.

### update

Update is called every frame, if the Component is enabled.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:222</a>

## Parameters

- `dt` [Number](#) the delta time in seconds it took to complete the last frame

### lateUpdate

LateUpdate is called every frame, if the Component is enabled.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:233</a>

### \_\_preload

\_\_preload is called before every onLoad. It is used to initialize the builtin components internally, to avoid checking whether onLoad is called before every public method calls. This method should be removed if script priority is supported.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:243</a>

### onLoad

When attaching to an active node or its node first activated. onLoad is always called before any start functions, this allows you to order initialization of scripts.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:254</a>

### start

Called before all scripts' update if the Component is enabled the first time. Usually used to initialize some logic which need to be called after all components' `onload` methods called. This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:267</a>

### onEnable

Called when this component becomes enabled and its node is active. This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:280</a>

### onDisable

Called when this component becomes disabled or its node becomes inactive. This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:290</a>

### onDestroy

Called when this component will be destroyed. This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

<b>meta</b>	<b>description</b>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:300</a>

#### onFocusInEditor

<b>meta</b>	<b>description</b>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:310</a>

#### onLostFocusInEditor

<b>meta</b>	<b>description</b>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:315</a>

#### resetInEditor

Called to initialize the component or node's properties when adding the component the first time or when the Reset command is used. This function is only called in editor.

<b>meta</b>	<b>description</b>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:320</a>

#### addComponent

Adds a component class to the node. You can also add component to node by passing in the name of the script.

<b>meta</b>	<b>description</b>
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:330</a>

## Parameters

- `typeOrClassName` [Function](#) | [String](#) the constructor or the class name of the component to add

## Examples

```
var sprite = node.addComponent(cc.Sprite);
var test = node.addComponent("Test");
```

### getComponent

Returns the component of supplied type if the node has one attached, null if it doesn't. You can also get component in the node by passing in the name of the script.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:348</a>

## Parameters

- `typeOrClassName` [Function](#) | [String](#)

## Examples

```
// get sprite component.
var sprite = node.getComponent(cc.Sprite);
// get custom test calss.
var test = node.getComponent("Test");
```

### getComponents

Returns all components of supplied Type in the node.

meta	description
Returns	<a href="#">Component[]</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:372</a>

## Parameters

- `typeOrClassName` [Function](#) | [String](#)

## Examples

```
var sprites = node.getComponents(cc.Sprite);
var tests = node.getComponents("Test");
```

### getComponentInChildren

Returns the component of supplied type in any of its children using depth first search.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:390</a>

#### Parameters

- `typeOrClassName` [Function](#) | [String](#)

## Examples

```
var sprite = node.getComponentInChildren(cc.Sprite);
var Test = node.getComponentInChildren("Test");
```

### getComponentsInChildren

Returns the components of supplied type in self or any of its children using depth first search.

meta	description
Returns	<a href="#">Component[]</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:408</a>

#### Parameters

- `typeOrClassName` [Function](#) | [String](#)

## Examples

```
var sprites = node.getComponentsInChildren(cc.Sprite);
var tests = node.getComponentsInChildren("Test");
```

## `_getLocalBounds`

If the component's bounding box is different from the node's, you can implement this method to supply a custom axis aligned bounding box (AABB), so the editor's scene view can perform hit test properly.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:428</a>

### Parameters

- `out_rect Rect` the Rect to receive the bounding box

### onRestore

`onRestore` is called after the user clicks the Reset item in the Inspector's context menu or performs an undo operation on this component.

If the component contains the "internal state", short for "temporary member variables which not included in its CCClass properties", then you may need to implement this function.

The editor will call the getset accessors of your component to record/restore the component's state for undo/redo operation. However, in extreme cases, it may not work well. Then you should implement this function to manually synchronize your component's "internal states" with its public properties. Once you implement this function, all the getset accessors of your component will not be called when the user performs an undo/redo operation. Which means that only the properties with default value will be recorded or restored by editor.

Similarly, the editor may fail to reset your component correctly in extreme cases. Then if you need to support the reset menu, you should manually synchronize your component's "internal states" with its properties in this function. Once you implement this function, all the getset accessors of your component will not be called during reset operation. Which means that only the properties with default value will be reset by editor.

This function is only called in editor mode.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:441</a>

## schedule

Schedules a custom selector.

If the selector is already scheduled, then the interval parameter will be updated without scheduling it again.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:535</a>

### Parameters

- `callback function` The callback function
- `interval Number` Tick interval in seconds. 0 means tick every frame.
- `repeat Number` The selector will be executed (repeat + 1) times, you can use cc.macro.REPEAT\_FOREVER for tick infinitely.
- `delay Number` The amount of time that the first tick will wait before execution.

### Examples

```
var timeCallback = function (dt) {
    cc.log("time: " + dt);
}
this.schedule(timeCallback, 1);
```

## scheduleOnce

Schedules a callback function that runs only once, with a delay of 0 or larger.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:572</a>

### Parameters

- `callback function` A function wrapped as a selector
- `delay Number` The amount of time that the first tick will wait before execution.

## Examples

```
var timeCallback = function (dt) {  
    cc.log("time: " + dt);  
}  
this.scheduleOnce(timeCallback, 2);
```

## unschedule

Unschedules a custom callback function.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:589</a>

## Parameters

- `callback_fn` [function](#) A function wrapped as a selector

## Examples

```
this.unschedule(_callback);
```

## unscheduleAllCallbacks

unschedule all scheduled callback functions: custom callback functions, and the 'update' callback function.

Actions are not affected by this method.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:605</a>

## Examples

```
this.unscheduleAllCallbacks();
```

## destroy

Destroy this Object, and release all its own references to other objects.

Actual object destruction will delayed until before rendering. From the next frame, this object is not usable any more. You can use `cc.isValid(obj)` to check whether the object is destroyed before accessing it.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:296</a>

## Examples

```
obj.destroy();
```

### \_destruct

Clear all references in the instance.

NOTE: this method will not clear the getter or setter functions which defined in the instance of CCObject. You can override the \_destruct method if you need, for example: \_destruct: function () { for (var key in this) { if (this.hasOwnProperty(key)) { switch (typeof this[key]) { case 'string': this[key] = ''; break; case 'object': case 'function': this[key] = null; break; } } }

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:430</a>

### \_onPreDestroy

Called before the object being destroyed.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:463</a>

### \_serialize

The customized serialization for this object. (Editor Only)

meta	description
Returns	<a href="#">object</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:488</a>

## Parameters

- `exporting Boolean`

`_deserialize`

Init this object from the custom serialized data.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:498</a>

## Parameters

- `data Object` the serialized json data
- `ctx _Deserializer`

# DragonBonesAsset Class

Extends [Asset](#)

Module: [dragonBones](#)

The skeleton data of dragonBones.

## Index

### Properties

- `dragonBonesJson string` See <http://developer.egret.com/cn/github/egret-docs/DB/dbLibs/dataFormat/index.html>
- `loaded Boolean` Whether the asset is loaded or not
- `nativeUrl String` Returns the url of this asset's native object, if none it will returns an empty string.
- `_native String` Serializable url for native asset.
- `_nativeAsset Object` The underlying native asset of this asset if one is available.
- `_uuid String`
- `_name String`
- `_objFlags Number`
- `name String` The name of the object.
- `isValid Boolean` Indicates whether the object is not yet destroyed.

### Methods

- `toString` Returns the asset's url.

- `serialize` 应 AssetDB 要求提供这个方法
- `createNode` Create a new node using this asset in the scene....
- `_setRawAsset` Set native file name for this asset.
- `destroy` Actual object destruction will delayed until before rendering.
- `_destruct` Clear all references in the instance.
- `_onPreDestroy` Called before the object being destroyed.
- `_serialize` The customized serialization for this object.
- `_deserialize` Init this object from the custom serialized data.

## Details

### Properties

`dragonBonesJson`

See <http://developer.egret.com/cn/github/egret-docs/DB/dbLibs/dataFormat/index.html>

meta	description
Type	<code>string</code>
Defined in	<a href="#">extensions/dragonbones/DragonBonesAsset.js:51</a>

`loaded`

Whether the asset is loaded or not

meta	description
Type	<code>Boolean</code>
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:57</a>

`nativeUrl`

Returns the url of this asset's native object, if none it will returns an empty string.

meta	description
Type	<code>String</code>

<b>meta</b>	<b>description</b>
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:70</a>

#### \_native

Serializable url for native asset.

<b>meta</b>	<b>description</b>
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:108</a>

#### \_nativeAsset

The underlying native asset of this asset if one is available. This property can be used to access additional details or functionality related to the asset. This property will be initialized by the loader if \_native is available.

<b>meta</b>	<b>description</b>
Type	<a href="#">Object</a>
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:116</a>

#### \_uuid

<b>meta</b>	<b>description</b>
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/assets/CCRawAsset.js:46</a>

### `_name`

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:76</a>

### `_objFlags`

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:83</a>

### `name`

The name of the object.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:243</a>

### Examples

```
obj.name = "New Obj";
```

### `isValid`

Indicates whether the object is not yet destroyed. (It will not be available after being destroyed)  
When an object's `destroy` is called, it is actually destroyed after the end of this frame.  
So `isValid` will return false from the next frame, while `isValid` in the current frame will still be true.  
If you want to determine whether the current frame has called `destroy`, use `cc.isValid(obj, true)`, but this is often caused by a particular logical requirements, which is not normally required.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:261</a>

## Examples

```
var node = new cc.Node();
cc.log(node.isValid);      // true
node.destroy();
cc.log(node.isValid);      // true, still valid in this frame
// after a frame...
cc.log(node.isValid);      // false, destroyed in the end of last frame
```

## Methods

### toString

Returns the asset's url.

The Asset object overrides the `toString()` method of the Object object. For Asset objects, the `toString()` method returns a string representation of the object. JavaScript calls the `toString()` method automatically when an asset is to be represented as a text value or when a texture is referred to in a string concatenation.

meta	description
Returns	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:165</a>

### serialize

应 AssetDB 要求提供这个方法

meta	description
Returns	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:179</a>

## createNode

Create a new node using this asset in the scene.

If this type of asset dont have its corresponding node type, this method should be null.

meta	description
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:190</a>

## Parameters

- `callback` [Function](#)
- `error` [String](#) null or the error info
- `node` [Object](#) the created node or null

## \_setRawAsset

Set native file name for this asset.

meta	description
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:205</a>

## Parameters

- `filename` [String](#)
- `inLibrary` [Boolean](#)

## destroy

Destroy this Object, and release all its own references to other objects.

Actual object destruction will delayed until before rendering. From the next frame, this object is not usable any more. You can use cc.isValid(obj) to check whether the object is destroyed before accessing it.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:296</a>

## Examples

```
obj.destroy();
```

### \_destruct

Clear all references in the instance.

NOTE: this method will not clear the getter or setter functions which defined in the instance of CCOObject. You can override the \_destruct method if you need, for example: \_destruct: function () { for (var key in this) { if (this.hasOwnProperty(key)) { switch (typeof this[key]) { case 'string': this[key] = ""; break; case 'object': case 'function': this[key] = null; break; } } }}

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:430</a>

### \_onPreDestroy

Called before the object being destroyed.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:463</a>

### \_serialize

The customized serialization for this object. (Editor Only)

meta	description
Returns	<a href="#">object</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:488</a>

## Parameters

- exporting [Boolean](#)

### \_deserialize

Init this object from the custom serialized data.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:498</a>

## Parameters

- `data Object` the serialized json data
- `ctx _Deserializer`

## DragonBonesAtlasAsset Class

Extends [Asset](#)

Module: [dragonBones](#)

The skeleton atlas data of dragonBones.

## Index

### Properties

- `atlasJson string`
- `texture Texture2D`
- `loaded Boolean` Whether the asset is loaded or not
- `nativeUrl String` Returns the url of this asset's native object, if none it will returns an empty string.
- `_native String` Serializable url for native asset.
- `_nativeAsset Object` The underlying native asset of this asset if one is available.
- `_uuid String`
- `_name String`
- `_objFlags Number`
- `name String` The name of the object.
- `isValid Boolean` Indicates whether the object is not yet destroyed.

### Methods

- `toString` Returns the asset's url.
- `serialize` 应 AssetDB 要求提供这个方法
- `createNode` Create a new node using this asset in the scene....
- `_setRawAsset` Set native file name for this asset.
- `destroy` Actual object destruction will delayed until before rendering.
- `_destruct` Clear all references in the instance.
- `_onPreDestroy` Called before the object being destroyed.
- `_serialize` The customized serialization for this object.
- `_deserialize` Init this object from the custom serialized data.

## Details

## Properties

atlasJson

meta	description
Type	<a href="#">string</a>
Defined in	<a href="#">extensions/dragonbones/DragonBonesAtlasAsset.js:51</a>

texture

meta	description
Type	<a href="#">Texture2D</a>
Defined in	<a href="#">extensions/dragonbones/DragonBonesAtlasAsset.js:70</a>

loaded

Whether the asset is loaded or not

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:57</a>

nativeUrl

Returns the url of this asset's native object, if none it will returns an empty string.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:70</a>

### \_native

Serializable url for native asset.

<b>meta</b>	<b>description</b>
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:108</a>

### \_nativeAsset

The underlying native asset of this asset if one is available. This property can be used to access additional details or functionality related to the asset. This property will be initialized by the loader if `_native` is available.

<b>meta</b>	<b>description</b>
Type	<a href="#">Object</a>
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:116</a>

### \_uuid

<b>meta</b>	<b>description</b>
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/assets/CCRawAsset.js:46</a>

### \_name

<b>meta</b>	<b>description</b>
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:76</a>

## \_objFlags

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:83</a>

## name

The name of the object.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:243</a>

## Examples

```
obj.name = "New Obj";
```

## isValid

Indicates whether the object is not yet destroyed. (It will not be available after being destroyed)  
When an object's `destroy` is called, it is actually destroyed after the end of this frame.  
So `isValid` will return false from the next frame, while `isValid` in the current frame will still be true.  
If you want to determine whether the current frame has called `destroy`, use `cc.isValid(obj, true)`, but this is often caused by a particular logical requirements, which is not normally required.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:261</a>

## Examples

```
var node = new cc.Node();
cc.log(node.isValid);    // true
node.destroy();
cc.log(node.isValid);    // true, still valid in this frame
// after a frame...
cc.log(node.isValid);    // false, destroyed in the end of last frame
```

## Methods

### toString

Returns the asset's url.

The Asset object overrides the `toString()` method of the Object object. For Asset objects, the `toString()` method returns a string representation of the object. JavaScript calls the `toString()` method automatically when an asset is to be represented as a text value or when a texture is referred to in a string concatenation.

meta	description
Returns	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:165</a>

### serialize

应 AssetDB 要求提供这个方法

meta	description
Returns	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:179</a>

### createNode

Create a new node using this asset in the scene.

If this type of asset dont have its corresponding node type, this method should be null.

meta	description
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:190</a>

### Parameters

- `callback` [Function](#)
- `error` [String](#) null or the error info
- `node` [Object](#) the created node or null

## \_setRawAsset

Set native file name for this asset.

meta	description
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:205</a>

## Parameters

- `filename` [String](#)
- `inLibrary` [Boolean](#)

## destroy

Destroy this Object, and release all its own references to other objects.

Actual object destruction will delayed until before rendering. From the next frame, this object is not usable any more. You can use cc.isValid(obj) to check whether the object is destroyed before accessing it.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:296</a>

## Examples

```
obj.destroy();
```

## \_destruct

Clear all references in the instance.

NOTE: this method will not clear the getter or setter functions which defined in the instance of CCObject. You can override the \_destruct method if you need, for example:

```
_destruct: function () { for (var key in this) { if (this.hasOwnProperty(key)) { switch (typeof this[key]) { case 'string': this[key] = ''; break; case 'object': case 'function': this[key] = null; break; } } }}
```

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:430</a>

### \_onPreDestroy

Called before the object being destroyed.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:463</a>

### \_serialize

The customized serialization for this object. (Editor Only)

meta	description
Returns	<a href="#">object</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:488</a>

Parameters

- `exporting Boolean`

### \_deserialize

Init this object from the custom serialized data.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:498</a>

Parameters

- `data Object` the serialized json data
- `ctx _Deserializer`

## EditBox Class

Extends [Component](#)

Module: [cc](#) Parent Module: [cc](#)

cc.EditBox is a component for inputing text, you can use it to gather small amounts of text from users.

## Index

### Properties

- `string` String Input string of EditBox.
- `backgroundImage` SpriteFrame The background image of EditBox.
- `returnType` EditBox.KeyboardReturnType The return key type of EditBox.
- `inputFlag` EditBox.InputFlag Set the input flags that are to be applied to the EditBox.
- `inputMode` EditBox.InputMode Set the input mode of the edit box.
- `fontSize` Number Font size of the input text.
- `lineHeight` Number Change the lineHeight of displayed text.
- `fontColor` Color Font color of the input text.
- `placeholder` String The display text of placeholder.
- `placeholderFontSize` Number The font size of placeholder.
- `placeholderFontColor` Color The font color of placeholder.
- `maxLength` Number The maximize input length of EditBox.
- `stayOnTop` Boolean The input is always visible and be on top of the game view (only useful on Web).
- `tabIndex` Number Set the tabIndex of the DOM input element (only useful on Web).
- `editingDidBegan` Component.EventHandler[] The event handler to be called when EditBox began to edit text.
- `textChanged` Component.EventHandler[] The event handler to be called when EditBox text changes.
- `editingDidEnded` Component.EventHandler[] The event handler to be called when EditBox edit ends.
- `editingReturn` Component.EventHandler[] The event handler to be called when return key is pressed.
- `_eventTargets` Array Register all related EventTargets,...
- `node` Node The node this component is attached to.
- `uuid` String The uuid for editor.
- `_enabled` Boolean
- `enabled` Boolean indicates whether this component is enabled or not.
- `enabledInHierarchy` Boolean indicates whether this component is enabled and its node is also active in the hierarchy.
- `_isOnLoadCalled` Number Returns a value which used to indicate the onLoad get called or not.
- `_name` String
- `_objFlags` Number
- `name` String The name of the object.
- `isValid` Boolean Indicates whether the object is not yet destroyed.

### Methods

- `setFocus` Let the EditBox get focus, only valid when stayOnTop is true.
- `isFocused` Determine whether EditBox is getting focus or not.
- `destroy` call the destroy method on this component or the associated node explicitly.

- `update` This is a lifecycle method.
- `lateUpdate` This is a lifecycle method.
- `__preload` `__preload` is called before every `onLoad`.
- `onLoad` When attaching to an active node or its node first activated.
- `start` Called before all scripts' update if the Component is enabled the first time.
- `onEnable` This is a lifecycle method.
- `onDisable` This is a lifecycle method.
- `onDestroy` This is a lifecycle method.
- `onFocusInEditor`
- `onLostFocusInEditor`
- `resetInEditor` Called to initialize the component or node's properties when adding the component the first time or when the Reset command is used.
- `addComponent` Adds a component class to the node.
- `getComponent` Returns the component of supplied type if the node has one attached, null if it doesn't....
- `getComponents` Returns all components of supplied Type in the node.
- `getComponentInChildren` Returns the component of supplied type in any of its children using depth first search.
- `getComponentsInChildren` Returns the components of supplied type in self or any of its children using depth first search.
- `_getLocalBounds` If the component's bounding box is different from the node's, you can implement this method to supply
- `onRestore` for undo/redo operation.
- `schedule` Schedules a custom selector....
- `scheduleOnce` Schedules a callback function that runs only once, with a delay of 0 or larger.
- `unschedule` Unschedules a custom callback function.
- `unscheduleAllCallbacks` unschedule all scheduled callback functions: custom callback functions, and the 'update' callback function....
- `_destruct` Clear all references in the instance.
- `_onPreDestroy` Called before the object being destroyed.
- `_serialize` The customized serialization for this object.
- `_deserialize` Init this object from the custom serialized data.

## Events

- `editing-did-began` Note: This event is emitted from the node to which the component belongs.
- `editing-did-ended` Note: This event is emitted from the node to which the component belongs.
- `text-changed` Note: This event is emitted from the node to which the component belongs.
- `editing-return` Note: This event is emitted from the node to which the component belongs.

## Details

## Properties

string

Input string of EditBox.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/components/editbox/CCEditBox.js:69</a>

backgroundImage

The background image of EditBox.

meta	description
Type	<a href="#">SpriteFrame</a>
Defined in	<a href="#">cocos2d/core/components/editbox/CCEditBox.js:91</a>

returnType

The return key type of EditBox. Note: it is meaningless for web platforms and desktop platforms.

meta	description
Type	<a href="#">EditBox.KeyboardReturnType</a>
Defined in	<a href="#">cocos2d/core/components/editbox/CCEditBox.js:105</a>

inputFlag

Set the input flags that are to be applied to the EditBox.

meta	description
Type	<a href="#">EditBox.InputFlag</a>
Defined in	<a href="#">cocos2d/core/components/editbox/CCEditBox.js:127</a>

### inputMode

Set the input mode of the edit box. If you pass ANY, it will create a multiline EditBox.

meta	description
Type	<a href="#">EditBox.InputMode</a>
Defined in	<a href="#">cocos2d/core/components/editbox/CCEditBox.js:144</a>

### fontSize

Font size of the input text.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/editbox/CCEditBox.js:164</a>

### lineHeight

Change the lineHeight of displayed text.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/editbox/CCEditBox.js:182</a>

## fontColor

Font color of the input text.

meta	description
Type	<a href="#">Color</a>
Defined in	<a href="#">cocos2d/core/components/editbox/CCEditBox.js:197</a>

## placeholder

The display text of placeholder.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/components/editbox/CCEditBox.js:216</a>

## placeholderFontSize

The font size of placeholder.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/editbox/CCEditBox.js:234</a>

## placeholderFontColor

The font color of placeholder.

meta	description
Type	<a href="#">Color</a>

<b>meta</b>	<b>description</b>
Defined in	<a href="#">cocos2d/core/components/editbox/CCEditBox.js:249</a>

### maxLength

The maximize input length of EditBox.

- If pass a value less than 0, it won't limit the input number of characters.
- If pass 0, it doesn't allow input any characters.

<b>meta</b>	<b>description</b>
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/editbox/CCEditBox.js:265</a>

### stayOnTop

The input is always visible and be on top of the game view (only useful on Web). !zh  
输入框总是可见，并且永远在游戏视图的上面（这个属性只有在 Web  
上面修改有意义） Note: only available on Web at the moment.

<b>meta</b>	<b>description</b>
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/editbox/CCEditBox.js:284</a>

### tabIndex

Set the tabIndex of the DOM input element (only useful on Web).

<b>meta</b>	<b>description</b>
Type	<a href="#">Number</a>

<b>meta</b>	<b>description</b>
Defined in	<a href="#">cocos2d/core/components/editbox/CCEditBox.js:302</a>

### editingDidBegan

The event handler to be called when EditBox began to edit text.

<b>meta</b>	<b>description</b>
Type	<a href="#">Component.EventHandler[]</a>
Defined in	<a href="#">cocos2d/core/components/editbox/CCEditBox.js:320</a>

### textChanged

The event handler to be called when EditBox text changes.

<b>meta</b>	<b>description</b>
Type	<a href="#">Component.EventHandler[]</a>
Defined in	<a href="#">cocos2d/core/components/editbox/CCEditBox.js:330</a>

### editingDidEnded

The event handler to be called when EditBox edit ends.

<b>meta</b>	<b>description</b>
Type	<a href="#">Component.EventHandler[]</a>
Defined in	<a href="#">cocos2d/core/components/editbox/CCEditBox.js:340</a>

### editingReturn

The event handler to be called when return key is pressed. Windows is not supported.

meta	description
Type	<a href="#">Component.EventHandler[]</a>
Defined in	<a href="#">cocos2d/core/components/editbox/CCEditBox.js:350</a>

### \_\_eventTargets

Register all related EventTargets, all event callbacks will be removed in  
`_onPreDestroy`

meta	description
Type	<a href="#">Array</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:61</a>

### node

The node this component is attached to. A component is always attached to a node.

meta	description
Type	<a href="#">Node</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:75</a>

### Examples

```
cc.log(comp.node);
```

### uuid

The uuid for editor.

meta	description
Type	<a href="#">String</a>

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:106</a>

## Examples

```
cc.log(comp.uuid);
```

### \_enabled

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:147</a>

### enabled

indicates whether this component is enabled or not.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:154</a>

## Examples

```
comp.enabled = true;
cc.log(comp.enabled);
```

### enabledInHierarchy

indicates whether this component is enabled and its node is also active in the hierarchy.

meta	description
Type	<a href="#">Boolean</a>

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:185</a>

## Examples

```
cc.log(comp.enabledInHierarchy);
```

### \_isOnLoadCalled

Returns a value which used to indicate the onLoad get called or not.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:201</a>

## Examples

```
cc.log(this._isOnLoadCalled > 0);
```

### \_name

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:76</a>

### \_objFlags

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:83</a>

## name

The name of the object.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:243</a>

## Examples

```
obj.name = "New Obj";
```

## isValid

Indicates whether the object is not yet destroyed. (It will not be available after being destroyed)

When an object's `destroy` is called, it is actually destroyed after the end of this frame. So `isValid` will return false from the next frame, while `isValid` in the current frame will still be true. If you want to determine whether the current frame has called `destroy`, use `cc.isValid(obj, true)`, but this is often caused by a particular logical requirements, which is not normally required.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:261</a>

## Examples

```
var node = new cc.Node();
cc.log(node.isValid);    // true
node.destroy();
cc.log(node.isValid);    // true, still valid in this frame
// after a frame...
cc.log(node.isValid);    // false, destroyed in the end of last frame
```

## Methods

### setFocus

Let the EditBox get focus, only valid when `stayOnTop` is true.

meta	description
Defined in	<a href="#">cocos2d/core/components/editbox/CCEditBox.js:624</a>

## isFocused

Determine whether EditBox is getting focus or not.

meta	description
Defined in	<a href="#">cocos2d/core/components/editbox/CCEditBox.js:636</a>

## destroy

If you don't need the EditBox and it isn't in any running Scene, you should call the destroy method on this component or the associated node explicitly. Otherwise, the created DOM element won't be removed from web page.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/editbox/CCEditBox.js:700</a>

## Examples

```
editbox.node.parent = null; // or editbox.node.removeFromParent(false);
// when you don't need editbox anymore
editbox.node.destroy();
```

## update

Update is called every frame, if the Component is enabled.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:222</a>

## Parameters

- `dt` [Number](#) the delta time in seconds it took to complete the last frame

## lateUpdate

LateUpdate is called every frame, if the Component is enabled.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:233</a>

## \_\_preload

`__preload` is called before every `onLoad`. It is used to initialize the builtin components internally, to avoid checking whether `onLoad` is called before every public method calls. This method should be removed if script priority is supported.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:243</a>

## onLoad

When attaching to an active node or its node first activated. `onLoad` is always called before any start functions, this allows you to order initialization of scripts.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:254</a>

## start

Called before all scripts' update if the Component is enabled the first time. Usually used to initialize some logic which need to be called after all components' `onload` methods called.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:267</a>

#### onEnable

Called when this component becomes enabled and its node is active.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:280</a>

#### onDisable

Called when this component becomes disabled or its node becomes inactive.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:290</a>

#### onDestroy

Called when this component will be destroyed.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:300</a>

## onFocusInEditor

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:310</a>

## onLostFocusInEditor

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:315</a>

## resetInEditor

Called to initialize the component or node's properties when adding the component the first time or when the Reset command is used. This function is only called in editor.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:320</a>

## addComponent

Adds a component class to the node. You can also add component to node by passing in the name of the script.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:330</a>

## Parameters

- `typeOrClassName` [Function](#) | [String](#) the constructor or the class name of the component to add

## Examples

```
var sprite = node.addComponent(cc.Sprite);
var test = node.addComponent("Test");
```

### getComponent

Returns the component of supplied type if the node has one attached, null if it doesn't.

You can also get component in the node by passing in the name of the script.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:348</a>

### Parameters

- `typeOrClassName` [Function](#) | [String](#)

## Examples

```
// get sprite component.
var sprite = node.getComponent(cc.Sprite);
// get custom test calss.
var test = node.getComponent("Test");
```

### getComponents

Returns all components of supplied Type in the node.

meta	description
Returns	<a href="#">Component[]</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:372</a>

### Parameters

- `typeOrClassName` [Function](#) | [String](#)

## Examples

```
var sprites = node.getComponents(cc.Sprite);
var tests = node.getComponents("Test");
```

### getComponentInChildren

Returns the component of supplied type in any of its children using depth first search.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:390</a>

## Parameters

- typeOrClassName [Function](#) | [String](#)

## Examples

```
var sprite = node.getComponentInChildren(cc.Sprite);
var Test = node.getComponentInChildren("Test");
```

### getComponentsInChildren

Returns the components of supplied type in self or any of its children using depth first search.

meta	description
Returns	<a href="#">Component[]</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:408</a>

## Parameters

- typeOrClassName [Function](#) | [String](#)

## Examples

```
var sprites = node.getComponentsInChildren(cc.Sprite);
var tests = node.getComponentsInChildren("Test");
```

## \_getLocalBounds

If the component's bounding box is different from the node's, you can implement this method to supply a custom axis aligned bounding box (AABB), so the editor's scene view can perform hit test properly.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:428</a>

### Parameters

- `out_rect` [Rect](#) the Rect to receive the bounding box

### onRestore

onRestore is called after the user clicks the Reset item in the Inspector's context menu or performs an undo operation on this component.

If the component contains the "internal state", short for "temporary member variables which not included in its CCClass properties", then you may need to implement this function.

The editor will call the getset accessors of your component to record/restore the component's state for undo/redo operation. However, in extreme cases, it may not work well. Then you should implement this function to manually synchronize your component's "internal states" with its public properties. Once you implement this function, all the getset accessors of your component will not be called when the user performs an undo/redo operation. Which means that only the properties with default value will be recorded or restored by editor.

Similarly, the editor may fail to reset your component correctly in extreme cases. Then if you need to support the reset menu, you should manually synchronize your component's "internal states" with its properties in this function. Once you implement this function, all the getset accessors of your component will not be called during reset operation. Which means that only the properties with default value will be reset by editor.

This function is only called in editor mode.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:441</a>

## schedule

Schedules a custom selector.

If the selector is already scheduled, then the interval parameter will be updated without scheduling it again.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:535</a>

## Parameters

- `callback function` The callback function
- `interval Number` Tick interval in seconds. 0 means tick every frame.
- `repeat Number` The selector will be executed (repeat + 1) times, you can use cc.macro.REPEAT\_FOREVER for tick infinitely.
- `delay Number` The amount of time that the first tick will wait before execution.

## Examples

```
var timeCallback = function (dt) {
    cc.log("time: " + dt);
}
this.schedule(timeCallback, 1);
```

## scheduleOnce

Schedules a callback function that runs only once, with a delay of 0 or larger.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:572</a>

## Parameters

- `callback function` A function wrapped as a selector

- `delay` [Number](#) The amount of time that the first tick will wait before execution.

### Examples

```
var timeCallback = function (dt) {
    cc.log("time: " + dt);
}
this.scheduleOnce(timeCallback, 2);
```

### unschedule

Unschedules a custom callback function.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:589</a>

### Parameters

- `callback_fn` [function](#) A function wrapped as a selector

### Examples

```
this.unschedule(_callback);
```

### unscheduleAllCallbacks

unschedule all scheduled callback functions: custom callback functions, and the 'update' callback function.

Actions are not affected by this method.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:605</a>

### Examples

```
this.unscheduleAllCallbacks();
```

### \_destruct

Clear all references in the instance.

NOTE: this method will not clear the getter or setter functions which defined in the instance of CCObject. You can override the `_destruct` method if you need, for

```
example: _destruct: function () { for (var key in this) { if (this.hasOwnProperty(key)) { switch (typeof this[key]) { case 'string': this[key] = ""; break; case 'object': case 'function': this[key] = null; break; } } }
```

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:430</a>

### \_onPreDestroy

Called before the object being destroyed.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:463</a>

### \_serialize

The customized serialization for this object. (Editor Only)

meta	description
Returns	<a href="#">object</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:488</a>

### Parameters

- exporting [Boolean](#)

### \_deserialize

Init this object from the custom serialized data.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:498</a>

## Parameters

- data [Object](#) the serialized json data
- ctx [\\_Deserializer](#)

## Events

### **editing-did-began Event**

Module: [cc](#) Parent Module: [cc](#)

Note: This event is emitted from the node to which the component belongs.

## Index

## Details

### **editing-did-ended Event**

Module: [cc](#) Parent Module: [cc](#)

Note: This event is emitted from the node to which the component belongs.

## Index

## Details

### **text-changed Event**

Module: [cc](#) Parent Module: [cc](#)

Note: This event is emitted from the node to which the component belongs.

## Index

## Details

### **editing-return Event**

Module: [cc](#) Parent Module: [cc](#)

Note: This event is emitted from the node to which the component belongs.

### **EqualToFrame Class**

Extends [ContainerStrategy](#)

Module: [decorator](#) Parent Module: [cc](#)

# Index

## Methods

- `preApply` Manipulation before applying the strategy
- `apply` Function to apply this strategy
- `postApply` Manipulation after applying the strategy

## Details

### *Methods*

#### `preApply`

Manipulation before applying the strategy

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCView.js:1031</a>

#### Parameters

- `view` [View](#) The target view

#### `apply`

Function to apply this strategy

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCView.js:1041</a>

#### Parameters

- `view` [View](#)
- `designedResolution` [Size](#)

#### `postApply`

Manipulation after applying the strategy

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCView.js:1052</a>

## Parameters

- `view` [View](#) The target view

## EqualToWindow Class

Extends [EqualToFrame](#)

Module: [decorator](#) Parent Module: [cc](#)

## Index

### Methods

- `preApply` Manipulation before applying the strategy
- `apply` Function to apply this strategy
- `postApply` Manipulation after applying the strategy

## Details

### Methods

#### preApply

Manipulation before applying the strategy

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCView.js:1031</a>

## Parameters

- `view` [View](#) The target view

#### apply

Function to apply this strategy

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCView.js:1041</a>

## Parameters

- `view` [View](#)
- `designedResolution` [Size](#)

## postApply

Manipulation after applying the strategy

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCView.js:1052</a>

## Parameters

- `view` [View](#) The target view

# Event Class

Defined in: <https://github.com/cocos-creator/engine/blob/4f734a806d1fd7c4073fb064fddc961384fe67af/cocos2d/core/event/event.js:32>

Module: [cc](#)

Base class of all kinds of events.

# Index

## Properties

- `type` String The name of the event (case-sensitive), e.g.
- `bubbles` Boolean Indicate whether the event bubbles up through the tree or not.
- `target` Object A reference to the target to which the event was originally dispatched.
- `currentTarget` Object A reference to the currently registered target for the event.
- `eventPhase` Number Indicates which phase of the event flow is currently being evaluated.
- `NO_TYPE` String Code for event without type.
- `TOUCH` String The type code of Touch event.
- `MOUSE` String The type code of Mouse event.

- **KEYBOARD** String The type code of Keyboard event.
- **ACCELERATION** String The type code of Acceleration event.
- **NONE** Number Events not currently dispatched are in this phase
- **CAPTURING\_PHASE** Number The capturing phase comprises the journey from the root to the last node before the event target's node
- **AT\_TARGET** Number The target phase comprises only the event target node
- **BUBBLING\_PHASE** Number The bubbling phase comprises any subsequent nodes encountered on the return trip to the root of the hierarchy

## Methods

- **constructor**
- **unuse** Reset the event for being stored in the object pool.
- **reuse** Reuse the event for being used again by the object pool.
- **stopPropagation** Stops propagation for current event.
- **stopPropagationImmediate** Stops propagation for current event immediately,...
- **isStopped** Checks whether the event has been stopped.
- **getCurrentTarget** note: It only be available when the event listener is associated with node.
- **getType** Gets the event type.

## Details

### *Properties*

#### type

The name of the event (case-sensitive), e.g. "click", "fire", or "submit".

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/event/event.js:44</a>

#### bubbles

Indicate whether the event bubbles up through the tree or not.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/event/event.js:52</a>

## target

A reference to the target to which the event was originally dispatched.

meta	description
Type	<a href="#">Object</a>
Defined in	<a href="#">cocos2d/core/event/event.js:60</a>

## currentTarget

A reference to the currently registered target for the event.

meta	description
Type	<a href="#">Object</a>
Defined in	<a href="#">cocos2d/core/event/event.js:68</a>

## eventPhase

Indicates which phase of the event flow is currently being evaluated. Returns an integer value represented by 4 constants:

- Event.NONE = 0
- Event.CAPTURING\_PHASE = 1
- Event.AT\_TARGET = 2
- Event.BUBBLING\_PHASE = 3 The phases are explained in the [section 3.1, Event dispatch and DOM event flow] (<http://www.w3.org/TR/DOM-Level-3-Events/#event-flow>), of the DOM Level 3 Events specification.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/event/event.js:76</a>

## NO\_TYPE

Code for event without type.

<b>meta</b>	<b>description</b>
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/event/event.js:193</a>

## TOUCH

The type code of Touch event.

<b>meta</b>	<b>description</b>
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/event/event.js:202</a>

## MOUSE

The type code of Mouse event.

<b>meta</b>	<b>description</b>
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/event/event.js:210</a>

## KEYBOARD

The type code of Keyboard event.

<b>meta</b>	<b>description</b>
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/event/event.js:218</a>

## ACCELERATION

The type code of Acceleration event.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/event/event.js:226</a>

## NONE

Events not currently dispatched are in this phase

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/event/event.js:236</a>

## CAPTURING\_PHASE

The capturing phase comprises the journey from the root to the last node before the event target's node see <http://www.w3.org/TR/DOM-Level-3-Events/#event-flow>

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/event/event.js:244</a>

## AT\_TARGET

The target phase comprises only the event target node see <http://www.w3.org/TR/DOM-Level-3-Events/#event-flow>

meta	description
Type	<a href="#">Number</a>

meta	description
Defined in	<a href="#">cocos2d/core/event/event.js:254</a>

## BUBBLING\_PHASE

The bubbling phase comprises any subsequent nodes encountered on the return trip to the root of the hierarchy see <http://www.w3.org/TR/DOM-Level-3-Events/#event-flow>

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/event/event.js:264</a>

## Methods

### constructor

meta	description
Defined in	<a href="#">cocos2d/core/event/event.js:38</a>

### Parameters

- type [String](#) The name of the event (case-sensitive), e.g. "click", "fire", or "submit"
- bubbles [Boolean](#) A boolean indicating whether the event bubbles up through the tree or not

### unuse

Reset the event for being stored in the object pool.

meta	description
Returns	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/event/event.js:111</a>

## reuse

Reuse the event for being used again by the object pool.

meta	description
Returns	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/event/event.js:126</a>

## stopPropagation

Stops propagation for current event.

meta	description
Defined in	<a href="#">cocos2d/core/event/event.js:137</a>

## stopPropagationImmediate

Stops propagation for current event immediately, the event won't even be dispatched to the listeners attached in the current target.

meta	description
Defined in	<a href="#">cocos2d/core/event/event.js:146</a>

## isStopped

Checks whether the event has been stopped.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/event/event.js:156</a>

## getCurrentTarget

Gets current target of the event

note: It only be available when the event listener is associated with node.  
It returns 0 when the listener is associated with fixed priority.

meta	description
Returns	<a href="#">Node</a>
Defined in	<a href="#">cocos2d/core/event/event.js:166</a>

## getType

Gets the event type.

meta	description
Returns	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/event/event.js:181</a>

# Event.EventAcceleration Class

Extends [Event](#)

Module: [cc](#) Parent Module: [cc](#)

The acceleration event

## Index

### Properties

- `type` String The name of the event (case-sensitive), e.g.
- `bubbles` Boolean Indicate whether the event bubbles up through the tree or not.
- `target` Object A reference to the target to which the event was originally dispatched.
- `currentTarget` Object A reference to the currently registered target for the event.
- `eventPhase` Number Indicates which phase of the event flow is currently being evaluated.

### Methods

- `constructor`

- [unuse](#) Reset the event for being stored in the object pool.
- [reuse](#) Reuse the event for being used again by the object pool.
- [stopPropagation](#) Stops propagation for current event.
- [stopPropagationImmediate](#) Stops propagation for current event immediately,...
- [isStopped](#) Checks whether the event has been stopped.
- [getCurrentTarget](#) note: It only be available when the event listener is associated with node.
- [getType](#) Gets the event type.

## Details

### *Properties*

type

The name of the event (case-sensitive), e.g. "click", "fire", or "submit".

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/event/event.js:44</a>

bubbles

Indicate whether the event bubbles up through the tree or not.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/event/event.js:52</a>

target

A reference to the target to which the event was originally dispatched.

meta	description
Type	<a href="#">Object</a>
Defined in	<a href="#">cocos2d/core/event/event.js:60</a>

## currentTarget

A reference to the currently registered target for the event.

meta	description
Type	<a href="#">Object</a>
Defined in	<a href="#">cocos2d/core/event/event.js:68</a>

## eventPhase

Indicates which phase of the event flow is currently being evaluated. Returns an integer value represented by 4 constants:

- Event.NONE = 0
- Event.CAPTURING\_PHASE = 1
- Event.AT\_TARGET = 2
- Event.BUBBLING\_PHASE = 3 The phases are explained in the [section 3.1, Event dispatch and DOM event flow] (<http://www.w3.org/TR/DOM-Level-3-Events/#event-flow>), of the DOM Level 3 Events specification.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/event/event.js:76</a>

## Methods

### constructor

meta	description
Defined in	<a href="#">cocos2d/core/event/event.js:38</a>

### Parameters

- type [String](#) The name of the event (case-sensitive), e.g. "click", "fire", or "submit"
- bubbles [Boolean](#) A boolean indicating whether the event bubbles up through the tree or not

## unuse

Reset the event for being stored in the object pool.

meta	description
Returns	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/event/event.js:111</a>

## reuse

Reuse the event for being used again by the object pool.

meta	description
Returns	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/event/event.js:126</a>

## stopPropagation

Stops propagation for current event.

meta	description
Defined in	<a href="#">cocos2d/core/event/event.js:137</a>

## stopPropagationImmediate

Stops propagation for current event immediately, the event won't even be dispatched to the listeners attached in the current target.

meta	description
Defined in	<a href="#">cocos2d/core/event/event.js:146</a>

## isStopped

Checks whether the event has been stopped.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/event/event.js:156</a>

### getCurrentTarget

Gets current target of the event

note: It only be available when the event listener is associated with node.  
It returns 0 when the listener is associated with fixed priority.

meta	description
Returns	<a href="#">Node</a>
Defined in	<a href="#">cocos2d/core/event/event.js:166</a>

### getType

Gets the event type.

meta	description
Returns	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/event/event.js:181</a>

## Event.EventCustom Class

Extends [Event](#)

Module: [cc](#)

The Custom event

## Index

## Properties

- `detail` Object A reference to the detailed data of the event
- `type` String The name of the event (case-sensitive), e.g.
- `bubbles` Boolean Indicate whether the event bubbles up through the tree or not.
- `target` Object A reference to the target to which the event was originally dispatched.
- `currentTarget` Object A reference to the currently registered target for the event.
- `eventPhase` Number Indicates which phase of the event flow is currently being evaluated.

## Methods

- `constructor`
- `setUserData` Sets user data
- `getUserData` Gets user data
- `getEventName` Gets event name
- `unuse` Reset the event for being stored in the object pool.
- `reuse` Reuse the event for being used again by the object pool.
- `stopPropagation` Stops propagation for current event.
- `stopPropagationImmediate` Stops propagation for current event immediately,...
- `isStopped` Checks whether the event has been stopped.
- `getCurrentTarget` note: It only be available when the event listener is associated with node.
- `getType` Gets the event type.

## Details

### Properties

#### detail

A reference to the detailed data of the event

meta	description
Type	<a href="#">Object</a>
Defined in	<a href="#">cocos2d/core/event/event.js:291</a>

#### type

The name of the event (case-sensitive), e.g. "click", "fire", or "submit".

meta	description
Type	<a href="#">String</a>

<b>meta</b>	<b>description</b>
Defined in	<a href="#">cocos2d/core/event/event.js:44</a>

### bubbles

Indicate whether the event bubbles up through the tree or not.

<b>meta</b>	<b>description</b>
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/event/event.js:52</a>

### target

A reference to the target to which the event was originally dispatched.

<b>meta</b>	<b>description</b>
Type	<a href="#">Object</a>
Defined in	<a href="#">cocos2d/core/event/event.js:60</a>

### currentTarget

A reference to the currently registered target for the event.

<b>meta</b>	<b>description</b>
Type	<a href="#">Object</a>
Defined in	<a href="#">cocos2d/core/event/event.js:68</a>

### eventPhase

Indicates which phase of the event flow is currently being evaluated. Returns an integer value represented by 4 constants:

- Event.NONE = 0
- Event.CAPTURING\_PHASE = 1
- Event.AT\_TARGET = 2
- Event.BUBBLING\_PHASE = 3 The phases are explained in the [section 3.1, Event dispatch and DOM event flow] (<http://www.w3.org/TR/DOM-Level-3-Events/#event-flow>), of the DOM Level 3 Events specification.

<b>meta</b>	<b>description</b>
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/event/event.js:76</a>

## Methods

constructor

<b>meta</b>	<b>description</b>
Defined in	<a href="#">cocos2d/core/event/event.js:283</a>

Parameters

- type [String](#) The name of the event (case-sensitive), e.g. "click", "fire", or "submit"
- bubbles [Boolean](#) A boolean indicating whether the event bubbles up through the tree or not

setUserData

Sets user data

<b>meta</b>	<b>description</b>
Defined in	<a href="#">cocos2d/core/event/event.js:304</a>

Parameters

- data Any

getUserData

Gets user data

<b>meta</b>	<b>description</b>
Returns	Any
Defined in	<a href="#">cocos2d/core/event/event.js:314</a>

getEventName

Gets event name

<b>meta</b>	<b>description</b>
Returns	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/event/event.js:324</a>

unuse

Reset the event for being stored in the object pool.

<b>meta</b>	<b>description</b>
Returns	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/event/event.js:111</a>

reuse

Reuse the event for being used again by the object pool.

<b>meta</b>	<b>description</b>
Returns	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/event/event.js:126</a>

## stopPropagation

Stops propagation for current event.

meta	description
Defined in	<a href="#">cocos2d/core/event/event.js:137</a>

## stopPropagationImmediate

Stops propagation for current event immediately, the event won't even be dispatched to the listeners attached in the current target.

meta	description
Defined in	<a href="#">cocos2d/core/event/event.js:146</a>

## isStopped

Checks whether the event has been stopped.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/event/event.js:156</a>

## getCurrentTarget

Gets current target of the event

note: It only be available when the event listener is associated with node.  
It returns 0 when the listener is associated with fixed priority.

meta	description
Returns	<a href="#">Node</a>
Defined in	<a href="#">cocos2d/core/event/event.js:166</a>

## getType

Gets the event type.

meta	description
Returns	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/event/event.js:181</a>

## Event.EventKeyboard Class

Extends [Event](#)

Module: [cc](#) Parent Module: [cc](#)

The keyboard event

## Index

### Properties

- `keyCode` Number The keyCode read-only property represents a system and implementation dependent numerical code identifying the unmodified value of the pressed key.
- `type` String The name of the event (case-sensitive), e.g.
- `bubbles` Boolean Indicate whether the event bubbles up through the tree or not.
- `target` Object A reference to the target to which the event was originally dispatched.
- `currentTarget` Object A reference to the currently registered target for the event.
- `eventPhase` Number Indicates which phase of the event flow is currently being evaluated.

### Methods

- `constructor`
- `unuse` Reset the event for being stored in the object pool.
- `reuse` Reuse the event for being used again by the object pool.
- `stopPropagation` Stops propagation for current event.
- `stopPropagationImmediate` Stops propagation for current event immediately,...
- `isStopped` Checks whether the event has been stopped.
- `getCurrentTarget` note: It only be available when the event listener is associated with node.
- `getType` Gets the event type.

## Details

## Properties

### keyCode

The keyCode read-only property represents a system and implementation dependent numerical code identifying the unmodified value of the pressed key. This is usually the decimal ASCII (RFC 20) or Windows 1252 code corresponding to the key. If the key can't be identified, this value is 0.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:555</a>

### type

The name of the event (case-sensitive), e.g. "click", "fire", or "submit".

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/event/event.js:44</a>

### bubbles

Indicate whether the event bubbles up through the tree or not.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/event/event.js:52</a>

### target

A reference to the target to which the event was originally dispatched.

meta	description
Type	<a href="#">Object</a>
Defined in	<a href="#">cocos2d/core/event/event.js:60</a>

### currentTarget

A reference to the currently registered target for the event.

meta	description
Type	<a href="#">Object</a>
Defined in	<a href="#">cocos2d/core/event/event.js:68</a>

### eventPhase

Indicates which phase of the event flow is currently being evaluated. Returns an integer value represented by 4 constants:

- Event.NONE = 0
- Event.CAPTURING\_PHASE = 1
- Event.AT\_TARGET = 2
- Event.BUBBLING\_PHASE = 3 The phases are explained in the [section 3.1, Event dispatch and DOM event flow] (<http://www.w3.org/TR/DOM-Level-3-Events/#event-flow>), of the DOM Level 3 Events specification.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/event/event.js:76</a>

## Methods

constructor

meta	description
Defined in	<a href="#">cocos2d/core/event/event.js:38</a>

Parameters

- type [String](#) The name of the event (case-sensitive), e.g. "click", "fire", or "submit"
- bubbles [Boolean](#) A boolean indicating whether the event bubbles up through the tree or not

unuse

Reset the event for being stored in the object pool.

meta	description
Returns	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/event/event.js:111</a>

reuse

Reuse the event for being used again by the object pool.

meta	description
Returns	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/event/event.js:126</a>

stopPropagation

Stops propagation for current event.

meta	description
Defined in	<a href="#">cocos2d/core/event/event.js:137</a>

### stopPropagationImmediate

Stops propagation for current event immediately, the event won't even be dispatched to the listeners attached in the current target.

meta	description
Defined in	<a href="#">cocos2d/core/event/event.js:146</a>

### isStopped

Checks whether the event has been stopped.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/event/event.js:156</a>

### getCurrentTarget

Gets current target of the event

note: It only be available when the event listener is associated with node.  
It returns 0 when the listener is associated with fixed priority.

meta	description
Returns	<a href="#">Node</a>
Defined in	<a href="#">cocos2d/core/event/event.js:166</a>

### getType

Gets the event type.

meta	description
Returns	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/event/event.js:181</a>

## Event.EventMouse Class

Extends [Event](#)

Module: [cc](#) Parent Module: [cc](#)

The mouse event

## Index

### Properties

- `NONE` Number The none event code of mouse event.
- `DOWN` Number The event type code of mouse down event.
- `UP` Number The event type code of mouse up event.
- `MOVE` Number The event type code of mouse move event.
- `SCROLL` Number The event type code of mouse scroll event.
- `BUTTON_LEFT` Number The tag of Mouse left button.
- `BUTTON_RIGHT` Number The tag of Mouse right button (The right button number is 2 on browser).
- `BUTTON_MIDDLE` Number The tag of Mouse middle button (The right button number is 1 on browser).
- `BUTTON_4` Number The tag of Mouse button 4.
- `BUTTON_5` Number The tag of Mouse button 5.
- `BUTTON_6` Number The tag of Mouse button 6.
- `BUTTON_7` Number The tag of Mouse button 7.
- `BUTTON_8` Number The tag of Mouse button 8.
- `type` String The name of the event (case-sensitive), e.g.
- `bubbles` Boolean Indicate whether the event bubbles up through the tree or not.
- `target` Object A reference to the target to which the event was originally dispatched.
- `currentTarget` Object A reference to the currently registered target for the event.
- `eventPhase` Number Indicates which phase of the event flow is currently being evaluated.

### Methods

- `setScrollData` Sets scroll data.
- `getScrollX` Returns the x axis scroll value.
- `getScrollY` Returns the y axis scroll value.
- `setLocation` Sets cursor location.
- `getLocation` Returns cursor location.
- `getLocationInView` Returns the current cursor location in screen coordinates.
- `getPreviousLocation` Returns the previous touch location.

- `getDelta` Returns the delta distance from the previous location to current location.
- `getDeltaX` Returns the X axis delta distance from the previous location to current location.
- `getDeltaY` Returns the Y axis delta distance from the previous location to current location.
- `setButton` Sets mouse button.
- `getButton` Returns mouse button.
- `getLocationX` Returns location X axis data.
- `getLocationY` Returns location Y axis data.
- `constructor`
- `unuse` Reset the event for being stored in the object pool.
- `reuse` Reuse the event for being used again by the object pool.
- `stopPropagation` Stops propagation for current event.
- `stopPropagationImmediate` Stops propagation for current event immediately,...
- `isStopped` Checks whether the event has been stopped.
- `getCurrentTarget` note: It only be available when the event listener is associated with node.
- `getType` Gets the event type.

## Details

### *Properties*

NONE

The none event code of mouse event.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:208</a>

DOWN

The event type code of mouse down event.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:216</a>

UP

The event type code of mouse up event.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:224</a>

## MOVE

The event type code of mouse move event.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:232</a>

## SCROLL

The event type code of mouse scroll event.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:240</a>

## BUTTON\_LEFT

The tag of Mouse left button.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:249</a>

## BUTTON\_RIGHT

The tag of Mouse right button (The right button number is 2 on browser).

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:258</a>

## BUTTON\_MIDDLE

The tag of Mouse middle button (The right button number is 1 on browser).

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:267</a>

## BUTTON\_4

The tag of Mouse button 4.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:276</a>

## BUTTON\_5

The tag of Mouse button 5.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:285</a>

## BUTTON\_6

The tag of Mouse button 6.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:294</a>

## BUTTON\_7

The tag of Mouse button 7.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:303</a>

## BUTTON\_8

The tag of Mouse button 8.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:312</a>

## type

The name of the event (case-sensitive), e.g. "click", "fire", or "submit".

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/event/event.js:44</a>

## bubbles

Indicate whether the event bubbles up through the tree or not.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/event/event.js:52</a>

## target

A reference to the target to which the event was originally dispatched.

meta	description
Type	<a href="#">Object</a>
Defined in	<a href="#">cocos2d/core/event/event.js:60</a>

## currentTarget

A reference to the currently registered target for the event.

meta	description
Type	<a href="#">Object</a>
Defined in	<a href="#">cocos2d/core/event/event.js:68</a>

## eventPhase

Indicates which phase of the event flow is currently being evaluated. Returns an integer value represented by 4 constants:

- Event.NONE = 0
- Event.CAPTURING\_PHASE = 1
- Event.AT\_TARGET = 2
- Event.BUBBLING\_PHASE = 3 The phases are explained in the [section 3.1, Event dispatch and DOM event flow] (<http://www.w3.org/TR/DOM-Level-3-Events/#event-flow>), of the DOM Level 3 Events specification.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/event/event.js:76</a>

## Methods

`setScrollData`

Sets scroll data.

meta	description
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:58</a>

Parameters

- `scrollX` [Number](#)
- `scrollY` [Number](#)

`getScrollX`

Returns the x axis scroll value.

meta	description
Returns	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:70</a>

`getScrollY`

Returns the y axis scroll value.

meta	description
Returns	<a href="#">Number</a>

meta	description
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:80</a>

### setLocation

Sets cursor location.

meta	description
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:90</a>

### Parameters

- x [Number](#)
- y [Number](#)

### getLocation

Returns cursor location.

meta	description
Returns	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:102</a>

### getLocationInView

Returns the current cursor location in screen coordinates.

meta	description
Returns	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:112</a>

### getPreviousLocation

Returns the previous touch location.

meta	description
Returns	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:127</a>

### getDelta

Returns the delta distance from the previous location to current location.

meta	description
Returns	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:137</a>

### getDeltaX

Returns the X axis delta distance from the previous location to current location.

meta	description
Returns	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:147</a>

### getDeltaY

Returns the Y axis delta distance from the previous location to current location.

meta	description
Returns	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:157</a>

## setButton

Sets mouse button.

meta	description
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:167</a>

Parameters

- **button** [Number](#)

## getButton

Returns mouse button.

meta	description
Returns	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:177</a>

## getLocationX

Returns location X axis data.

meta	description
Returns	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:187</a>

## getLocationY

Returns location Y axis data.

meta	description
Returns	<a href="#">Number</a>

meta	description
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:197</a>

constructor

meta	description
Defined in	<a href="#">cocos2d/core/event/event.js:38</a>

Parameters

- type [String](#) The name of the event (case-sensitive), e.g. "click", "fire", or "submit"
- bubbles [Boolean](#) A boolean indicating whether the event bubbles up through the tree or not

unuse

Reset the event for being stored in the object pool.

meta	description
Returns	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/event/event.js:111</a>

reuse

Reuse the event for being used again by the object pool.

meta	description
Returns	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/event/event.js:126</a>

stopPropagation

Stops propagation for current event.

meta	description
Defined in	<a href="#">cocos2d/core/event/event.js:137</a>

### stopPropagationImmediate

Stops propagation for current event immediately, the event won't even be dispatched to the listeners attached in the current target.

meta	description
Defined in	<a href="#">cocos2d/core/event/event.js:146</a>

### isStopped

Checks whether the event has been stopped.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/event/event.js:156</a>

### getCurrentTarget

Gets current target of the event

note: It only be available when the event listener is associated with node.  
It returns 0 when the listener is associated with fixed priority.

meta	description
Returns	<a href="#">Node</a>
Defined in	<a href="#">cocos2d/core/event/event.js:166</a>

### getType

Gets the event type.

meta	description
Returns	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/event/event.js:181</a>

## Event.EventTouch Class

Extends [Event](#)

Module: [cc](#) Parent Module: [cc](#)

The touch event

## Index

### Properties

- `touch` Touch The current touch object
- `type` String The name of the event (case-sensitive), e.g.
- `bubbles` Boolean Indicate whether the event bubbles up through the tree or not.
- `target` Object A reference to the target to which the event was originally dispatched.
- `currentTarget` Object A reference to the currently registered target for the event.
- `eventPhase` Number Indicates which phase of the event flow is currently being evaluated.

### Methods

- `constructor`
- `getEventCode` Returns event code.
- `getTouches` Returns touches of event.
- `setLocation` Sets touch location.
- `getLocation` Returns touch location.
- `getLocationInView` Returns the current touch location in screen coordinates.
- `getPreviousLocation` Returns the previous touch location.
- `getStartLocation` Returns the start touch location.
- `getID` Returns the id of cc.Touch.
- `getDelta` Returns the delta distance from the previous location to current location.
- `getDeltaX` Returns the X axis delta distance from the previous location to current location.
- `getDeltaY` Returns the Y axis delta distance from the previous location to current location.
- `getLocationX` Returns location X axis data.
- `getLocationY` Returns location Y axis data.
- `unuse` Reset the event for being stored in the object pool.
- `reuse` Reuse the event for being used again by the object pool.
- `stopPropagation` Stops propagation for current event.
- `stopPropagationImmediate` Stops propagation for current event immediately,...
- `isStopped` Checks whether the event has been stopped.

- `getCurrentTarget` note: It only be available when the event listener is associated with node.
- `getType` Gets the event type.

## Details

### *Properties*

**touch**

The current touch object

meta	description
Type	<a href="#">Touch</a>
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:337</a>

**type**

The name of the event (case-sensitive), e.g. "click", "fire", or "submit".

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/event/event.js:44</a>

**bubbles**

Indicate whether the event bubbles up through the tree or not.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/event/event.js:52</a>

**target**

A reference to the target to which the event was originally dispatched.

meta	description
Type	<a href="#">Object</a>
Defined in	<a href="#">cocos2d/core/event/event.js:60</a>

### currentTarget

A reference to the currently registered target for the event.

meta	description
Type	<a href="#">Object</a>
Defined in	<a href="#">cocos2d/core/event/event.js:68</a>

### eventPhase

Indicates which phase of the event flow is currently being evaluated. Returns an integer value represented by 4 constants:

- Event.NONE = 0
- Event.CAPTURING\_PHASE = 1
- Event.AT\_TARGET = 2
- Event.BUBBLING\_PHASE = 3 The phases are explained in the [section 3.1, Event dispatch and DOM event flow] (<http://www.w3.org/TR/DOM-Level-3-Events/#event-flow>), of the DOM Level 3 Events specification.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/event/event.js:76</a>

## Methods

constructor

meta	description
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:328</a>

Parameters

- touchArr [Array](#) The array of the touches
- bubbles [Boolean](#) A boolean indicating whether the event bubbles up through the tree or not

getEventCode

Returns event code.

meta	description
Returns	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:352</a>

getTouches

Returns touches of event.

meta	description
Returns	<a href="#">Array</a>
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:362</a>

setLocation

Sets touch location.

meta	description
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:380</a>

#### Parameters

- x [Number](#)
- y [Number](#)

#### getLocation

Returns touch location.

meta	description
Returns	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:391</a>

#### getLocationInView

Returns the current touch location in screen coordinates.

meta	description
Returns	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:401</a>

#### getPreviousLocation

Returns the previous touch location.

meta	description
Returns	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:411</a>

## getStartLocation

Returns the start touch location.

meta	description
Returns	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:421</a>

## getID

Returns the id of cc.Touch.

meta	description
Returns	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:431</a>

## getDelta

Returns the delta distance from the previous location to current location.

meta	description
Returns	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:441</a>

## getDeltaX

Returns the X axis delta distance from the previous location to current location.

meta	description
Returns	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:451</a>

## getDeltaY

Returns the Y axis delta distance from the previous location to current location.

meta	description
Returns	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:461</a>

## getLocationX

Returns location X axis data.

meta	description
Returns	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:471</a>

## getLocationY

Returns location Y axis data.

meta	description
Returns	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/event-manager/CCEvent.js:481</a>

## unuse

Reset the event for being stored in the object pool.

meta	description
Returns	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/event/event.js:111</a>

## reuse

Reuse the event for being used again by the object pool.

meta	description
Returns	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/event/event.js:126</a>

## stopPropagation

Stops propagation for current event.

meta	description
Defined in	<a href="#">cocos2d/core/event/event.js:137</a>

## stopPropagationImmediate

Stops propagation for current event immediately, the event won't even be dispatched to the listeners attached in the current target.

meta	description
Defined in	<a href="#">cocos2d/core/event/event.js:146</a>

## isStopped

Checks whether the event has been stopped.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/event/event.js:156</a>

## getCurrentTarget

Gets current target of the event

note: It only be available when the event listener is associated with node.  
It returns 0 when the listener is associated with fixed priority.

meta	description
Returns	<a href="#">Node</a>
Defined in	<a href="#">cocos2d/core/event/event.js:166</a>

## getType

Gets the event type.

meta	description
Returns	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/event/event.js:181</a>

# EventTarget Class

Defined in: <https://github.com/cocos-creator/engine/blob/4f734a806d1fd7c4073fb064fddc961384fe67af/cocos2d/core/event/event-target.js:35>

Module: [cc](#)

EventTarget is an object to which an event is dispatched when something has occurred. Entity are the most common event targets, but other objects can be event targets too.

Event targets are an important part of the Fireball event model. The event target serves as the focal point for how events flow through the scene graph. When an event such as a mouse click or a keypress occurs, Fireball dispatches an event object into the event flow from the root of the hierarchy. The event object then makes its way through the scene graph until it reaches the event target, at which point it begins its return trip through the scene graph. This round-trip journey to the event target is conceptually divided into three phases:

- The capture phase comprises the journey from the root to the last node before the event target's node
- The target phase comprises only the event target node

- The bubbling phase comprises any subsequent nodes encountered on the return trip to the root of the tree See also: <http://www.w3.org/TR/DOM-Level-3-Events/#event-flow>

Event targets can implement the following methods:

- `_getCapturingTargets`
- `_getBubblingTargets`

## Index

### Methods

- `hasEventListener` Checks whether the EventTarget object has any callback registered for a specific type of event.
- `on` Register an callback of a specific event type on the EventTarget.
- `off` Removes the listeners previously registered with the same type, callback, target and or useCapture,...
- `targetOff` Removes all callbacks previously registered with the same target (passed as parameter).
- `once` Register an callback of a specific event type on the EventTarget,...
- `emit` Trigger an event directly with the event name and necessary arguments.
- `dispatchEvent` Send an event with the event object.

## Details

### Methods

#### `hasEventListener`

Checks whether the EventTarget object has any callback registered for a specific type of event.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/event/event-target.js:68</a>

#### Parameters

- `type String` The type of event.

#### `on`

Register an callback of a specific event type on the EventTarget. This type of event should be triggered via `emit`.

meta	description
Returns	<a href="#">Function</a>
Defined in	<a href="#">cocos2d/core/event/event-target.js:76</a>

## Parameters

- type [String](#) A string representing the event type to listen for.
- callback [Function](#) The callback that will be invoked when the event is dispatched.
- The callback is ignored if it is a duplicate (the callbacks are unique).
- arg1 Any arg1
- arg2 Any arg2
- arg3 Any arg3
- arg4 Any arg4
- arg5 Any arg5
- target [Object](#) The target (this object) to invoke the callback, can be null

## Examples

```
eventTarget.on('fire', function (event) {
    cc.log("fire in the hole");
}, node);
```

## off

Removes the listeners previously registered with the same type, callback, target and/or useCapture, if only type is passed as parameter, all listeners registered with that type will be removed.

meta	description
Defined in	<a href="#">cocos2d/core/event/event-target.js:117</a>

## Parameters

- type [String](#) A string representing the event type being removed.
- callback [Function](#) The callback to remove.
- target [Object](#) The target (this object) to invoke the callback, if it's not given, only callback without target will be removed

## Examples

```
// register fire eventListener
var callback = eventTarget.on('fire', function (event) {
    cc.log("fire in the hole");
```

```

}, target);
// remove fire event listener
eventTarget.off('fire', callback, target);
// remove all fire event listeners
eventTarget.off('fire');

```

## targetOff

Removes all callbacks previously registered with the same target (passed as parameter). This is not for removing all listeners in the current event target, and this is not for removing all listeners the target parameter have registered. It's only for removing all listeners (callback and target couple) registered on the current event target by the target parameter.

meta	description
Defined in	<a href="#">cocos2d/core/event/event-target.js:151</a>

## Parameters

- target [Object](#) The target to be searched for all related listeners

## once

Register an callback of a specific event type on the EventTarget, the callback will remove itself after the first time it is triggered.

meta	description
Defined in	<a href="#">cocos2d/core/event/event-target.js:164</a>

## Parameters

- type [String](#) A string representing the event type to listen for.
- callback [Function](#) The callback that will be invoked when the event is dispatched.
- The callback is ignored if it is a duplicate (the callbacks are unique).
- arg1 Any arg1
- arg2 Any arg2
- arg3 Any arg3
- arg4 Any arg4
- arg5 Any arg5
- target [Object](#) The target (this object) to invoke the callback, can be null

## Examples

```

eventTarget.once('fire', function (event) {
    cc.log("this is the callback and will be invoked only once");
}, node);

```

## emit

Trigger an event directly with the event name and necessary arguments.

meta	description
Defined in	<a href="#">cocos2d/core/event/event-target.js:201</a>

### Parameters

- type [String](#) event type
- arg1 Any First argument
- arg2 Any Second argument
- arg3 Any Third argument
- arg4 Any Fourth argument
- arg5 Any Fifth argument

### Examples

```
eventTarget.emit('fire', event);
eventTarget.emit('fire', message, emitter);
```

## dispatchEvent

Send an event with the event object.

meta	description
Defined in	<a href="#">cocos2d/core/event/event-target.js:221</a>

### Parameters

- event [Event](#)

## FiniteTimeAction Class

Extends [Action](#)

Module: [cc](#)

Base class actions that do have a finite time duration.

Possible actions:

- An action with a duration of 0 seconds.
- An action with a duration of 35.5 seconds.

Infinite time actions are valid

## Index

### Methods

- `getDuration` get duration of the action.
- `setDuration` set duration of the action.
- `reverse` Returns a reversed action.
- `clone` to copy object with deep copy.
- `isDone` return true if the action has finished.
- `getTarget` get the target.
- `setTarget` The action will modify the target properties.
- `getOriginalTarget` get the original target.
- `getTag` get tag number.
- `setTag` set tag number.

## Details

### Methods

#### getDuration

get duration of the action. (seconds).

meta	description
Returns	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/actions/CCAction.js:205</a>

#### setDuration

set duration of the action. (seconds).

meta	description
Defined in	<a href="#">cocos2d/actions/CCAction.js:215</a>

### Parameters

- `duration` [Number](#)

## reverse

Returns a reversed action.

For example:

- The action will be x coordinates of 0 move to 100.
- The reversed action will be x of 100 move to 0.
- Will be rewritten

meta	description
Returns	Null
Defined in	<a href="#">cocos2d/actions/CCAction.js:225</a>

## clone

to copy object with deep copy. returns a clone of action.

meta	description
Returns	<a href="#">FiniteTimeAction</a>
Defined in	<a href="#">cocos2d/actions/CCAction.js:241</a>

## isDone

return true if the action has finished.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/actions/CCAction.js:70</a>

## getTarget

get the target.

meta	description
Returns	<a href="#">Node</a>
Defined in	<a href="#">cocos2d/actions/CCAction.js:102</a>

## setTarget

The action will modify the target properties.

meta	description
Defined in	<a href="#">cocos2d/actions/CCAction.js:112</a>

## Parameters

- target [Node](#)

## getOriginalTarget

get the original target.

meta	description
Returns	<a href="#">Node</a>
Defined in	<a href="#">cocos2d/actions/CCAction.js:122</a>

## getTag

get tag number.

meta	description
Returns	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/actions/CCAction.js:139</a>

## setTag

set tag number.

meta	description
Defined in	<a href="#">cocos2d/actions/CCAction.js:149</a>

### Parameters

- `tag` [Number](#)

## Font Class

Extends [Asset](#)

Module: [cc](#)

Class for Font handling.

## Index

### Properties

- `loaded` Boolean Whether the asset is loaded or not
- `nativeUrl` String Returns the url of this asset's native object, if none it will returns an empty string.
- `_native` String Serializable url for native asset.
- `_nativeAsset` Object The underlying native asset of this asset if one is available.
- `_uuid` String
- `_name` String
- `_objFlags` Number
- `name` String The name of the object.
- `isValid` Boolean Indicates whether the object is not yet destroyed.

### Methods

- `toString` Returns the asset's url.
- `serialize` 应 AssetDB 要求提供这个方法
- `createNode` Create a new node using this asset in the scene....
- `_setRawAsset` Set native file name for this asset.
- `destroy` Actual object destruction will delayed until before rendering.
- `_destruct` Clear all references in the instance.
- `_onPreDestroy` Called before the object being destroyed.
- `_serialize` The customized serialization for this object.
- `_deserialize` Init this object from the custom serialized data.

## Details

### Properties

loaded

Whether the asset is loaded or not

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:57</a>

nativeUrl

Returns the url of this asset's native object, if none it will returns an empty string.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:70</a>

\_native

Serializable url for native asset.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:108</a>

\_nativeAsset

The underlying native asset of this asset if one is available. This property can be used to access additional details or functionality related to the asset. This property will be initialized by the loader if \_native is available.

meta	description
Type	<a href="#">Object</a>
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:116</a>

\_uuid

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/assets/CCRawAsset.js:46</a>

\_name

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:76</a>

\_objFlags

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:83</a>

name

The name of the object.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:243</a>

## Examples

```
obj.name = "New Obj";
```

### isValid

Indicates whether the object is not yet destroyed. (It will not be available after being destroyed)  
When an object's `destroy` is called, it is actually destroyed after the end of this frame.  
So `isValid` will return false from the next frame, while `isValid` in the current frame will still be true.  
If you want to determine whether the current frame has called `destroy`, use `cc.isValid(obj, true)`, but this is often caused by a particular logical requirements, which is not normally required.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:261</a>

## Examples

```
var node = new cc.Node();
cc.log(node.isValid);      // true
node.destroy();
cc.log(node.isValid);      // true, still valid in this frame
// after a frame...
cc.log(node.isValid);      // false, destroyed in the end of last frame
```

## Methods

### toString

Returns the asset's url.

The Asset object overrides the `toString()` method of the Object object. For Asset objects, the `toString()` method returns a string representation of the object. JavaScript calls the `toString()` method automatically when an asset is to be represented as a text value or when a texture is referred to in a string concatenation.

meta	description
Returns	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:165</a>

serialize

应 AssetDB 要求提供这个方法

meta	description
Returns	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:179</a>

createNode

Create a new node using this asset in the scene.

If this type of asset dont have its corresponding node type, this method should be null.

meta	description
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:190</a>

Parameters

- callback [Function](#)
- error [String](#) null or the error info
- node [Object](#) the created node or null

\_setRawAsset

Set native file name for this asset.

meta	description
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:205</a>

## Parameters

- `filename` [String](#)
- `inLibrary` [Boolean](#)

## destroy

Destroy this Object, and release all its own references to other objects.

Actual object destruction will delayed until before rendering. From the next frame, this object is not usable any more. You can use `cc.isValid(obj)` to check whether the object is destroyed before accessing it.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:296</a>

## Examples

```
obj.destroy();
```

```
_destruct
```

Clear all references in the instance.

NOTE: this method will not clear the getter or setter functions which defined in the instance of CCObject. You can override the `_destruct` method if you need, for example:

```
_destruct: function () { for (var key in this) { if (this.hasOwnProperty(key)) { switch (typeof this[key]) { case 'string': this[key] = ""; break; case 'object': case 'function': this[key] = null; break; } } }}
```

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:430</a>

```
_onPreDestroy
```

Called before the object being destroyed.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:463</a>

## \_serialize

The customized serialization for this object. (Editor Only)

meta	description
Returns	<a href="#">object</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:488</a>

Parameters

- `exporting Boolean`

## \_deserialize

Init this object from the custom serialized data.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:498</a>

Parameters

- `data Object` the serialized json data
- `ctx _Deserializer`

# Graphics Class

Extends [Component](#)  
Module: [cc](#)

## Index

Properties

- `lineWidth Number` Current line width.
- `lineJoin Graphics.LineJoin` lineJoin determines how two connecting segments (of lines, arcs or curves) with non-zero lengths in a shape are joined together.
- `lineCap Graphics.LineCap` lineCap determines how the end points of every line are drawn.
- `strokeColor Color` stroke color
- `fillColor Color` fill color

- `miterLimit` Number Sets the miter limit ratio
- `__eventTargets` Array Register all related EventTargets,...
- `node` Node The node this component is attached to.
- `uuid` String The uuid for editor.
- `_enabled` Boolean
- `enabled` Boolean indicates whether this component is enabled or not.
- `enabledInHierarchy` Boolean indicates whether this component is enabled and its node is also active in the hierarchy.
- `_isOnLoadCalled` Number Returns a value which used to indicate the onLoad get called or not.
- `_name` String
- `_objFlags` Number
- `name` String The name of the object.
- `isValid` Boolean Indicates whether the object is not yet destroyed.

## Methods

- `moveTo` Move path start point to (x,y).
- `lineTo` Adds a straight line to the path
- `bezierCurveTo` Adds a cubic Bézier curve to the path
- `quadraticCurveTo` Adds a quadratic Bézier curve to the path
- `arc` Adds an arc to the path which is centered at (cx, cy) position with radius r starting at startAngle and ending at endAngle going in the given direction by counterclockwise (defaulting to false).
- `ellipse` Adds an ellipse to the path.
- `circle` Adds a circle to the path.
- `rect` Adds an rectangle to the path.
- `roundRect` Adds an round corner rectangle to the path.
- `fillRect` Draws a filled rectangle.
- `clear` Erasing any previously drawn content.
- `close` Causes the point of the pen to move back to the start of the current path.
- `stroke` Strokes the current or given path with the current stroke style.
- `fill` Fills the current or given path with the current fill style.
- `update` This is a lifecycle method.
- `lateUpdate` This is a lifecycle method.
- `__preload` \_\_preload is called before every onLoad.
- `onLoad` When attaching to an active node or its node first activated.
- `start` Called before all scripts' update if the Component is enabled the first time.
- `onEnable` This is a lifecycle method.
- `onDisable` This is a lifecycle method.
- `onDestroy` This is a lifecycle method.
- `onFocusInEditor`
- `onLostFocusInEditor`
- `resetInEditor` Called to initialize the component or node's properties when adding the component the first time or when the Reset command is used.
- `addComponent` Adds a component class to the node.
- `getComponent` Returns the component of supplied type if the node has one attached, null if it doesn't....
- `getComponents` Returns all components of supplied Type in the node.
- `getComponentInChildren` Returns the component of supplied type in any of its children using depth first search.
- `getComponentsInChildren` Returns the components of supplied type in self or any of its children using depth first search.

- `_getLocalBounds` If the component's bounding box is different from the node's, you can implement this method to supply
- `onRestore` for undo/redo operation.
- `schedule` Schedules a custom selector....
- `scheduleOnce` Schedules a callback function that runs only once, with a delay of 0 or larger.
- `unschedule` Unschedules a custom callback function.
- `unscheduleAllCallbacks` unschedule all scheduled callback functions: custom callback functions, and the 'update' callback function....
- `destroy` Actual object destruction will delayed until before rendering.
- `_destruct` Clear all references in the instance.
- `_onPreDestroy` Called before the object being destroyed.
- `_serialize` The customized serialization for this object.
- `_deserialize` Init this object from the custom serialized data.

## Details

### *Properties*

#### lineWidth

Current line width.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/graphics/graphics.js:61</a>

#### lineJoin

lineJoin determines how two connecting segments (of lines, arcs or curves) with non-zero lengths in a shape are joined together.

meta	description
Type	<a href="#">Graphics.LineJoin</a>
Defined in	<a href="#">cocos2d/core/graphics/graphics.js:79</a>

#### lineCap

lineCap determines how the end points of every line are drawn.

meta	description
Type	<a href="#">Graphics.LineCap</a>
Defined in	<a href="#">cocos2d/core/graphics/graphics.js:98</a>

strokeColor

stroke color

meta	description
Type	<a href="#">Color</a>
Defined in	<a href="#">cocos2d/core/graphics/graphics.js:117</a>

fillColor

fill color

meta	description
Type	<a href="#">Color</a>
Defined in	<a href="#">cocos2d/core/graphics/graphics.js:134</a>

miterLimit

Sets the miter limit ratio

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/graphics/graphics.js:151</a>

## \_\_eventTargets

Register all related EventTargets, all event callbacks will be removed in \_onPreDestroy

meta	description
Type	<a href="#">Array</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:61</a>

## node

The node this component is attached to. A component is always attached to a node.

meta	description
Type	<a href="#">Node</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:75</a>

## Examples

```
cc.log(comp.node);
```

## uuid

The uuid for editor.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:106</a>

## Examples

```
cc.log(comp.uuid);
```

### \_enabled

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:147</a>

### **enabled**

indicates whether this component is enabled or not.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:154</a>

### Examples

```
comp.enabled = true;  
cc.log(comp.enabled);
```

### **enabledInHierarchy**

indicates whether this component is enabled and its node is also active in the hierarchy.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:185</a>

### Examples

```
cc.log(comp.enabledInHierarchy);
```

### \_isOnLoadCalled

Returns a value which used to indicate the onLoad get called or not.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:201</a>

## Examples

```
cc.log(this.\_isOnLoadCalled > 0);
```

`_name`

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:76</a>

`_objFlags`

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:83</a>

`name`

The name of the object.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:243</a>

## Examples

```
obj.name = "New Obj";
```

### isValid

Indicates whether the object is not yet destroyed. (It will not be available after being destroyed)  
When an object's `destroy` is called, it is actually destroyed after the end of this frame.  
So `isValid` will return false from the next frame, while `isValid` in the current frame will still be true.  
If you want to determine whether the current frame has called `destroy`, use `cc.isValid(obj, true)`, but this is often caused by a particular logical requirements, which is not normally required.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:261</a>

## Examples

```
var node = new cc.Node();
cc.log(node.isValid);      // true
node.destroy();
cc.log(node.isValid);      // true, still valid in this frame
// after a frame...
cc.log(node.isValid);      // false, destroyed in the end of last frame
```

## Methods

### moveTo

Move path start point to (x,y).

meta	description
Defined in	<a href="#">cocos2d/core/graphics/graphics.js:212</a>

### Parameters

- x [Number](#) The x axis of the coordinate for the end point.
- y [Number](#) The y axis of the coordinate for the end point.

### lineTo

Adds a straight line to the path

meta	description
Defined in	<a href="#">cocos2d/core/graphics/graphics.js:227</a>

#### Parameters

- `x` [Number](#) The x axis of the coordinate for the end point.
- `y` [Number](#) The y axis of the coordinate for the end point.

#### bezierCurveTo

Adds a cubic Bézier curve to the path

meta	description
Defined in	<a href="#">cocos2d/core/graphics/graphics.js:242</a>

#### Parameters

- `c1x` [Number](#) The x axis of the coordinate for the first control point.
- `c1y` [Number](#) The y axis of the coordinate for first control point.
- `c2x` [Number](#) The x axis of the coordinate for the second control point.
- `c2y` [Number](#) The y axis of the coordinate for the second control point.
- `x` [Number](#) The x axis of the coordinate for the end point.
- `y` [Number](#) The y axis of the coordinate for the end point.

#### quadraticCurveTo

Adds a quadratic Bézier curve to the path

meta	description
Defined in	<a href="#">cocos2d/core/graphics/graphics.js:257</a>

#### Parameters

- `cx` [Number](#) The x axis of the coordinate for the control point.
- `cy` [Number](#) The y axis of the coordinate for the control point.
- `x` [Number](#) The x axis of the coordinate for the end point.
- `y` [Number](#) The y axis of the coordinate for the end point.

## arc

Adds an arc to the path which is centered at (cx, cy) position with radius r starting at startAngle and ending at endAngle going in the given direction by counterclockwise (defaulting to false).

meta	description
Defined in	<a href="#">cocos2d/core/graphics/graphics.js:270</a>

### Parameters

- cx [Number](#) The x axis of the coordinate for the center point.
- cy [Number](#) The y axis of the coordinate for the center point.
- r [Number](#) The arc's radius.
- startAngle [Number](#) The angle at which the arc starts, measured clockwise from the positive x axis and expressed in radians.
- endAngle [Number](#) The angle at which the arc ends, measured clockwise from the positive x axis and expressed in radians.
- counterclockwise [Boolean](#) An optional Boolean which, if true, causes the arc to be drawn counter-clockwise between the two angles. By default it is drawn clockwise.

## ellipse

Adds an ellipse to the path.

meta	description
Defined in	<a href="#">cocos2d/core/graphics/graphics.js:285</a>

### Parameters

- cx [Number](#) The x axis of the coordinate for the center point.
- cy [Number](#) The y axis of the coordinate for the center point.
- rx [Number](#) The ellipse's x-axis radius.
- ry [Number](#) The ellipse's y-axis radius.

## circle

Adds an circle to the path.

meta	description
Defined in	<a href="#">cocos2d/core/graphics/graphics.js:298</a>

## Parameters

- `cx Number` The x axis of the coordinate for the center point.
- `cy Number` The y axis of the coordinate for the center point.
- `r Number` The circle's radius.

## rect

Adds an rectangle to the path.

meta	description
Defined in	<a href="#">cocos2d/core/graphics/graphics.js:310</a>

## Parameters

- `x Number` The x axis of the coordinate for the rectangle starting point.
- `y Number` The y axis of the coordinate for the rectangle starting point.
- `w Number` The rectangle's width.
- `h Number` The rectangle's height.

## roundRect

Adds an round corner rectangle to the path.

meta	description
Defined in	<a href="#">cocos2d/core/graphics/graphics.js:323</a>

## Parameters

- `x Number` The x axis of the coordinate for the rectangle starting point.
- `y Number` The y axis of the coordinate for the rectangle starting point.
- `w Number` The rectangles width.
- `h Number` The rectangle's height.
- `r Number` The radius of the rectangle.

## fillRect

Draws a filled rectangle.

meta	description
Defined in	<a href="#">cocos2d/core/graphics/graphics.js:337</a>

#### Parameters

- `x` [Number](#) The x axis of the coordinate for the rectangle starting point.
- `y` [Number](#) The y axis of the coordinate for the rectangle starting point.
- `w` [Number](#) The rectangle's width.
- `h` [Number](#) The rectangle's height.

#### clear

Erasing any previously drawn content.

meta	description
Defined in	<a href="#">cocos2d/core/graphics/graphics.js:351</a>

#### Parameters

- `clean` [Boolean](#) Whether to clean the graphics inner cache.

#### close

Causes the point of the pen to move back to the start of the current path. It tries to add a straight line from the current point to the start.

meta	description
Defined in	<a href="#">cocos2d/core/graphics/graphics.js:361</a>

#### stroke

Strokes the current or given path with the current stroke style.

meta	description
Defined in	<a href="#">cocos2d/core/graphics/graphics.js:370</a>

## fill

Fills the current or given path with the current fill style.

meta	description
Defined in	<a href="#">cocos2d/core/graphics/graphics.js:379</a>

## update

Update is called every frame, if the Component is enabled.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:222</a>

## Parameters

- `dt Number` the delta time in seconds it took to complete the last frame

## lateUpdate

LateUpdate is called every frame, if the Component is enabled.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:233</a>

## \_\_preload

`__preload` is called before every `onLoad`. It is used to initialize the builtin components internally, to avoid checking whether `onLoad` is called before every public method calls. This method should be removed if script priority is supported.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:243</a>

## onLoad

When attaching to an active node or its node first activated. onLoad is always called before any start functions, this allows you to order initialization of scripts.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:254</a>

## start

Called before all scripts' update if the Component is enabled the first time. Usually used to initialize some logic which need to be called after all components' onload methods called.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:267</a>

## onEnable

Called when this component becomes enabled and its node is active.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:280</a>

## onDisable

Called when this component becomes disabled or its node becomes inactive.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:290</a>

## onDestroy

Called when this component will be destroyed.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:300</a>

## onFocusInEditor

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:310</a>

## onLostFocusInEditor

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:315</a>

## resetInEditor

Called to initialize the component or node's properties when adding the component the first time or when the Reset command is used. This function is only called in editor.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:320</a>

## addComponent

Adds a component class to the node. You can also add component to node by passing in the name of the script.

meta	description
Returns	<a href="#">Component</a>

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:330</a>

## Parameters

- `typeOrClassName` [Function](#) | [String](#) the constructor or the class name of the component to add

## Examples

```
var sprite = node.addComponent(cc.Sprite);
var test = node.addComponent("Test");
```

## getComponent

Returns the component of supplied type if the node has one attached, null if it doesn't. You can also get component in the node by passing in the name of the script.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:348</a>

## Parameters

- `typeOrClassName` [Function](#) | [String](#)

## Examples

```
// get sprite component.
var sprite = node.getComponent(cc.Sprite);
// get custom test calss.
var test = node.getComponent("Test");
```

## getComponents

Returns all components of supplied Type in the node.

meta	description
Returns	<a href="#">Component[]</a>

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:372</a>

## Parameters

- typeOrClassName [Function](#) | [String](#)

## Examples

```
var sprites = node.getComponents(cc.Sprite);
var tests = node.getComponents("Test");
```

### getComponentInChildren

Returns the component of supplied type in any of its children using depth first search.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:390</a>

## Parameters

- typeOrClassName [Function](#) | [String](#)

## Examples

```
var sprite = node.getComponentInChildren(cc.Sprite);
var Test = node.getComponentInChildren("Test");
```

### getComponentsInChildren

Returns the components of supplied type in self or any of its children using depth first search.

meta	description
Returns	<a href="#">Component[]</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:408</a>

## Parameters

- `typeOrClassName` [Function](#) | [String](#)

## Examples

```
var sprites = node.getComponentsInChildren(cc.Sprite);
var tests = node.getComponentsInChildren("Test");
```

## \_getLocalBounds

If the component's bounding box is different from the node's, you can implement this method to supply a custom axis aligned bounding box (AABB), so the editor's scene view can perform hit test properly.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:428</a>

## Parameters

- `out_rect` [Rect](#) the Rect to receive the bounding box

## onRestore

onRestore is called after the user clicks the Reset item in the Inspector's context menu or performs an undo operation on this component.

If the component contains the "internal state", short for "temporary member variables which not included in its CCClass properties", then you may need to implement this function.

The editor will call the getset accessors of your component to record/restore the component's state for undo/redo operation. However, in extreme cases, it may not work well. Then you should implement this function to manually synchronize your component's "internal states" with its public properties. Once you implement this function, all the getset accessors of your component will not be called when the user performs an undo/redo operation. Which means that only the properties with default value will be recorded or restored by editor.

Similarly, the editor may fail to reset your component correctly in extreme cases. Then if you need to support the reset menu, you should manually synchronize your component's "internal states" with its properties in this function. Once you implement this function, all the getset accessors of your component

will not be called during reset operation. Which means that only the properties with default value will be reset by editor.

This function is only called in editor mode.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:441</a>

## schedule

Schedules a custom selector.

If the selector is already scheduled, then the interval parameter will be updated without scheduling it again.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:535</a>

### Parameters

- `callback` [function](#) The callback function
- `interval` [Number](#) Tick interval in seconds. 0 means tick every frame.
- `repeat` [Number](#) The selector will be executed (repeat + 1) times, you can use `cc.macro.REPEAT_FOREVER` for tick infinitely.
- `delay` [Number](#) The amount of time that the first tick will wait before execution.

## Examples

```
var timeCallback = function (dt) {
    cc.log("time: " + dt);
}
this.schedule(timeCallback, 1);
```

## scheduleOnce

Schedules a callback function that runs only once, with a delay of 0 or larger.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:572</a>

### Parameters

- `callback` [function](#) A function wrapped as a selector

- **delay** [Number](#) The amount of time that the first tick will wait before execution.

### Examples

```
var timeCallback = function (dt) {
    cc.log("time: " + dt);
}
this.scheduleOnce(timeCallback, 2);
```

### unschedule

Unschedules a custom callback function.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:589</a>

### Parameters

- **callback\_fn** [function](#) A function wrapped as a selector

### Examples

```
this.unschedule(_callback);
```

### unscheduleAllCallbacks

unschedule all scheduled callback functions: custom callback functions, and the 'update' callback function.

Actions are not affected by this method.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:605</a>

### Examples

```
this.unscheduleAllCallbacks();
```

### destroy

Destroy this Object, and release all its own references to other objects.

Actual object destruction will delayed until before rendering. From the next frame, this object is not usable any more. You can use cc.isValid(obj) to check whether the object is destroyed before accessing it.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:296</a>

## Examples

```
obj.destroy();
```

### \_destruct

Clear all references in the instance.

NOTE: this method will not clear the getter or setter functions which defined in the instance of CCObject. You can override the \_destruct method if you need, for example: \_destruct: function () { for (var key in this) { if (this.hasOwnProperty(key)) { switch (typeof this[key]) { case 'string': this[key] = ''; break; case 'object': case 'function': this[key] = null; break; } } }

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:430</a>

### \_onPreDestroy

Called before the object being destroyed.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:463</a>

### \_serialize

The customized serialization for this object. (Editor Only)

meta	description
Returns	<a href="#">object</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:488</a>

## Parameters

- `exporting Boolean`

### \_deserialize

Init this object from the custom serialized data.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:498</a>

## Parameters

- `data Object` the serialized json data
- `ctx _Deserializer`

# Intersection Class

Module: [cc](#)

Intersection helper class

## Index

### Methods

- `lineLine` Test line and line
- `lineRect` Test line and rect
- `linePolygon` Test line and polygon
- `rectRect` Test rect and rect
- `rectPolygon` Test rect and polygon
- `polygonPolygon` Test polygon and polygon
- `circleCircle` Test circle and circle
- `polygonCircle` Test polygon and circle
- `pointInPolygon` Test whether the point is in the polygon
- `pointLineDistance` Calculate the distance of point to line.

## Details

## Methods

### lineLine

Test line and line

meta	description
Returns	<a href="#">boolean</a>
Defined in	<a href="#">cocos2d/core/collider/CCIntersection.js:37</a>

### Parameters

- a1 [Vec2](#) The start point of the first line
- a2 [Vec2](#) The end point of the first line
- b1 [Vec2](#) The start point of the second line
- b2 [Vec2](#) The end point of the second line

### lineRect

Test line and rect

meta	description
Returns	<a href="#">boolean</a>
Defined in	<a href="#">cocos2d/core/collider/CCIntersection.js:68</a>

### Parameters

- a1 [Vec2](#) The start point of the line
- a2 [Vec2](#) The end point of the line
- b [Rect](#) The rect

### linePolygon

Test line and polygon

meta	description
Returns	<a href="#">boolean</a>

meta	description
Defined in	<a href="#">cocos2d/core/collider/CCIntersection.js:100</a>

#### Parameters

- a [Vec2](#) The start point of the line
- a2 [Vec2](#) The end point of the line
- b [Vec2\[\]](#) The polygon, a set of points

#### rectRect

Test rect and rect

meta	description
Returns	<a href="#">boolean</a>
Defined in	<a href="#">cocos2d/core/collider/CCIntersection.js:125</a>

#### Parameters

- a [Rect](#) The first rect
- b [Rect](#) The second rect

#### rectPolygon

Test rect and polygon

meta	description
Returns	<a href="#">boolean</a>
Defined in	<a href="#">cocos2d/core/collider/CCIntersection.js:155</a>

#### Parameters

- a [Rect](#) The rect
- b [Vec2\[\]](#) The polygon, a set of points

## `polygonPolygon`

Test polygon and polygon

meta	description
Returns	<a href="#">boolean</a>
Defined in	<a href="#">cocos2d/core/collider/CCIntersection.js:207</a>

Parameters

- a [Vec2\[\]](#) The first polygon, a set of points
- b [Vec2\[\]](#) The second polygon, a set of points

## `circleCircle`

Test circle and circle

meta	description
Returns	<a href="#">boolean</a>
Defined in	<a href="#">cocos2d/core/collider/CCIntersection.js:246</a>

Parameters

- a [Object](#) Object contains position and radius
- b [Object](#) Object contains position and radius

## `polygonCircle`

Test polygon and circle

meta	description
Returns	<a href="#">boolean</a>
Defined in	<a href="#">cocos2d/core/collider/CCIntersection.js:263</a>

## Parameters

- `polygon` [Vec2\[\]](#) The Polygon, a set of points
- `circle` [Object](#) Object contains position and radius

## pointInPolygon

Test whether the point is in the polygon

meta	description
Returns	<a href="#">boolean</a>
Defined in	<a href="#">cocos2d/core/collider/CCIntersection.js:292</a>

## Parameters

- `point` [Vec2](#) The point
- `polygon` [Vec2\[\]](#) The polygon, a set of points

## pointLineDistance

Calculate the distance of point to line.

meta	description
Returns	<a href="#">boolean</a>
Defined in	<a href="#">cocos2d/core/collider/CCIntersection.js:322</a>

## Parameters

- `point` [Vec2](#) The point
- `start` [Vec2](#) The start point of line
- `end` [Vec2](#) The end point of line
- `isSegment` [boolean](#) whether this line is a segment

# Joint Class

Extends [Component](#)

Defined in: <https://github.com/cocos-creator/engine/blob/4f734a806d1fd7c4073fb064fddc961384fe67af/cocos2d/core/physics/joint/CCJoint.js:32>

Module: [cc](#) Parent Module: [cc](#)

Base class for joints to connect rigidbody.

## Index

### Properties

- `anchor` Vec2 The anchor of the rigidbody.
- `connectedAnchor` Vec2 The anchor of the connected rigidbody.
- `connectedBody` RigidBody The rigidbody to which the other end of the joint is attached.
- `collideConnected` Boolean Should the two rigid bodies connected with this joint collide with each other?
- `__eventTargets` Array Register all related EventTargets,...
- `node` Node The node this component is attached to.
- `uuid` String The uuid for editor.
- `_enabled` Boolean
- `enabled` Boolean indicates whether this component is enabled or not.
- `enabledInHierarchy` Boolean indicates whether this component is enabled and its node is also active in the hierarchy.
- `_isOnLoadCalled` Number Returns a value which used to indicate the onLoad get called or not.
- `_name` String
- `_objFlags` Number
- `name` String The name of the object.
- `isValid` Boolean Indicates whether the object is not yet destroyed.

### Methods

- `apply` Apply current changes to joint, this will regenerate inner box2d joint.
- `getWorldAnchor` Get the anchor point on rigidbody in world coordinates.
- `getWorldConnectedAnchor` Get the anchor point on connected rigidbody in world coordinates.
- `getReactionForce` Gets the reaction force of the joint.
- `getReactionTorque` Gets the reaction torque of the joint.
- `update` This is a lifecycle method.
- `lateUpdate` This is a lifecycle method.
- `__preload` \_\_preload is called before every onLoad.
- `onLoad` When attaching to an active node or its node first activated.
- `start` Called before all scripts' update if the Component is enabled the first time.
- `onEnable` This is a lifecycle method.
- `onDisable` This is a lifecycle method.
- `onDestroy` This is a lifecycle method.
- `onFocusInEditor`
- `onLostFocusInEditor`
- `resetInEditor` Called to initialize the component or node's properties when adding the component the first time or when the Reset command is used.
- `addComponent` Adds a component class to the node.

- `getComponent` Returns the component of supplied type if the node has one attached, null if it doesn't....
- `getComponents` Returns all components of supplied Type in the node.
- `getComponentInChildren` Returns the component of supplied type in any of its children using depth first search.
- `getComponentsInChildren` Returns the components of supplied type in self or any of its children using depth first search.
- `_getLocalBounds` If the component's bounding box is different from the node's, you can implement this method to supply
- `onRestore` for undo/redo operation.
- `schedule` Schedules a custom selector....
- `scheduleOnce` Schedules a callback function that runs only once, with a delay of 0 or larger.
- `unschedule` Unschedules a custom callback function.
- `unscheduleAllCallbacks` unschedule all scheduled callback functions: custom callback functions, and the 'update' callback function....
- `destroy` Actual object destruction will delayed until before rendering.
- `_destruct` Clear all references in the instance.
- `_onPreDestroy` Called before the object being destroyed.
- `_serialize` The customized serialization for this object.
- `_deserialize` Init this object from the custom serialized data.

## Details

### *Properties*

#### anchor

The anchor of the rigidbody.

meta	description
Type	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCJoint.js:49</a>

#### connectedAnchor

The anchor of the connected rigidbody.

meta	description
Type	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCJoint.js:61</a>

## connectedBody

The rigidbody to which the other end of the joint is attached.

meta	description
Type	<a href="#">RigidBody</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCJoint.js:74</a>

## collideConnected

Should the two rigid bodies connected with this joint collide with each other?

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCJoint.js:88</a>

## \_\_eventTargets

Register all related EventTargets, all event callbacks will be removed in `_onPreDestroy`

meta	description
Type	<a href="#">Array</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:61</a>

## node

The node this component is attached to. A component is always attached to a node.

meta	description
Type	<a href="#">Node</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:75</a>

## Examples

```
cc.log(comp.node);
```

### uuid

The uuid for editor.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:106</a>

## Examples

```
cc.log(comp.uuid);
```

### \_enabled

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:147</a>

### enabled

indicates whether this component is enabled or not.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:154</a>

## Examples

```
comp.enabled = true;  
cc.log(comp.enabled);
```

## `enabledInHierarchy`

indicates whether this component is enabled and its node is also active in the hierarchy.

<b>meta</b>	<b>description</b>
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:185</a>

## Examples

```
cc.log(comp.enabledInHierarchy);
```

### `_isOnLoadCalled`

Returns a value which used to indicate the onLoad get called or not.

<b>meta</b>	<b>description</b>
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:201</a>

## Examples

```
cc.log(this._isOnLoadCalled > 0);
```

### `_name`

<b>meta</b>	<b>description</b>
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:76</a>

## \_objFlags

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:83</a>

## name

The name of the object.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:243</a>

## Examples

```
obj.name = "New Obj";
```

## isValid

Indicates whether the object is not yet destroyed. (It will not be available after being destroyed)  
When an object's `destroy` is called, it is actually destroyed after the end of this frame.  
So `isValid` will return false from the next frame, while `isValid` in the current frame will still be true.  
If you want to determine whether the current frame has called `destroy`, use `cc.isValid(obj, true)`, but this is often caused by a particular logical requirements, which is not normally required.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:261</a>

## Examples

```
var node = new cc.Node();
cc.log(node.isValid);      // true
node.destroy();
cc.log(node.isValid);      // true, still valid in this frame
// after a frame...
cc.log(node.isValid);      // false, destroyed in the end of last frame
```

## Methods

### apply

Apply current changes to joint, this will regenerate inner box2d joint.

meta	description
Defined in	<a href="#">cocos2d/core/physics/joint/CCJoint.js:115</a>

### getWorldAnchor

Get the anchor point on rigidbody in world coordinates.

meta	description
Returns	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCJoint.js:127</a>

### getWorldConnectedAnchor

Get the anchor point on connected rigidbody in world coordinates.

meta	description
Returns	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCJoint.js:143</a>

### getReactionForce

Gets the reaction force of the joint.

meta	description
Returns	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCJoint.js:159</a>

## Parameters

- `timeStep` [Number](#) The time to calculate the reaction force for.

`getReactionTorque`

Gets the reaction torque of the joint.

meta	description
Returns	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCJoint.js:176</a>

## Parameters

- `timeStep` [Number](#) The time to calculate the reaction torque for.

`update`

Update is called every frame, if the Component is enabled.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:222</a>

## Parameters

- `dt` [Number](#) the delta time in seconds it took to complete the last frame

`lateUpdate`

LateUpdate is called every frame, if the Component is enabled.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:233</a>

## \_\_preload

\_\_preload is called before every onLoad. It is used to initialize the builtin components internally, to avoid checking whether onLoad is called before every public method calls. This method should be removed if script priority is supported.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:243</a>

## onLoad

When attaching to an active node or its node first activated. onLoad is always called before any start functions, this allows you to order initialization of scripts.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:254</a>

## start

Called before all scripts' update if the Component is enabled the first time. Usually used to initialize some logic which need to be called after all components' onload methods called.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:267</a>

## onEnable

Called when this component becomes enabled and its node is active.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:280</a>

### onDisable

Called when this component becomes disabled or its node becomes inactive.  
This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:290</a>

### onDestroy

Called when this component will be destroyed.  
This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:300</a>

### onFocusInEditor

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:310</a>

### onLostFocusInEditor

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:315</a>

### resetInEditor

Called to initialize the component or node's properties when adding the component the first time or when the Reset command is used. This function is only called in editor.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:320</a>

### addComponent

Adds a component class to the node. You can also add component to node by passing in the name of the script.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:330</a>

#### Parameters

- typeOrClassName [Function](#) | [String](#) the constructor or the class name of the component to add

#### Examples

```
var sprite = node.addComponent(cc.Sprite);
var test = node.addComponent("Test");
```

### getComponent

Returns the component of supplied type if the node has one attached, null if it doesn't. You can also get component in the node by passing in the name of the script.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:348</a>

#### Parameters

- typeOrClassName [Function](#) | [String](#)

#### Examples

```
// get sprite component.
```

```
var sprite = node.getComponent(cc.Sprite);
// get custom test class.
var test = node.getComponent("Test");
```

## getComponents

Returns all components of supplied Type in the node.

meta	description
Returns	<a href="#">Component[]</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:372</a>

### Parameters

- typeOrClassName [Function](#) | [String](#)

## Examples

```
var sprites = node.getComponents(cc.Sprite);
var tests = node.getComponents("Test");
```

## getComponentInChildren

Returns the component of supplied type in any of its children using depth first search.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:390</a>

### Parameters

- typeOrClassName [Function](#) | [String](#)

## Examples

```
var sprite = node.getComponentInChildren(cc.Sprite);
var Test = node.getComponentInChildren("Test");
```

## getComponentsInChildren

Returns the components of supplied type in self or any of its children using depth first search.

meta	description
Returns	<a href="#">Component[]</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:408</a>

## Parameters

- typeOrClassName [Function](#) | [String](#)

## Examples

```
var sprites = node.getComponentsInChildren(cc.Sprite);
var tests = node.getComponentsInChildren("Test");
```

## \_getLocalBounds

If the component's bounding box is different from the node's, you can implement this method to supply a custom axis aligned bounding box (AABB), so the editor's scene view can perform hit test properly.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:428</a>

## Parameters

- out\_rect [Rect](#) the Rect to receive the bounding box

## onRestore

onRestore is called after the user clicks the Reset item in the Inspector's context menu or performs an undo operation on this component.

If the component contains the "internal state", short for "temporary member variables which not included in its CCClass properties", then you may need to implement this function.

The editor will call the getset accessors of your component to record/restore the component's state for undo/redo operation. However, in extreme cases, it may not work well. Then you should implement this function to manually synchronize your component's "internal states" with its public properties. Once you implement this function, all the getset accessors of your component will not be called when the user performs an undo/redo operation. Which means that only the properties with default

value  
will be recorded or restored by editor.

Similarly, the editor may failed to reset your component correctly in extreme cases. Then if you need to support the reset menu, you should manually synchronize your component's "internal states" with its properties in this function. Once you implement this function, all the getset accessors of your component will not be called during reset operation. Which means that only the properties with default value will be reset by editor.

This function is only called in editor mode.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:441</a>

## schedule

Schedules a custom selector.

If the selector is already scheduled, then the interval parameter will be updated without scheduling it again.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:535</a>

## Parameters

- `callback function` The callback function
- `interval Number` Tick interval in seconds. 0 means tick every frame.
- `repeat Number` The selector will be executed (repeat + 1) times, you can use `cc.macro.REPEAT_FOREVER` for tick infinitely.
- `delay Number` The amount of time that the first tick will wait before execution.

## Examples

```
var timeCallback = function (dt) {
    cc.log("time: " + dt);
}
this.schedule(timeCallback, 1);
```

## scheduleOnce

Schedules a callback function that runs only once, with a delay of 0 or larger.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:572</a>

## Parameters

- `callback` [function](#) A function wrapped as a selector
- `delay` [Number](#) The amount of time that the first tick will wait before execution.

## Examples

```
var timeCallback = function (dt) {
    cc.log("time: " + dt);
}
this.scheduleOnce(timeCallback, 2);
```

## unschedule

Unschedules a custom callback function.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:589</a>

## Parameters

- `callback_fn` [function](#) A function wrapped as a selector

## Examples

```
this.unschedule(_callback);
```

## unscheduleAllCallbacks

unschedule all scheduled callback functions: custom callback functions, and the 'update' callback function.

Actions are not affected by this method.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:605</a>

## Examples

```
this.unscheduleAllCallbacks();
```

### destroy

Destroy this Object, and release all its own references to other objects.

Actual object destruction will delayed until before rendering. From the next frame, this object is not usable any more. You can use cc.isValid(obj) to check whether the object is destroyed before accessing it.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:296</a>

## Examples

```
obj.destroy();
```

### \_destruct

Clear all references in the instance.

NOTE: this method will not clear the getter or setter functions which defined in the instance of CCObject. You can override the \_destruct method if you need, for example: \_destruct: function () { for (var key in this) { if (this.hasOwnProperty(key)) { switch (typeof this[key]) { case 'string': this[key] = ''; break; case 'object': case 'function': this[key] = null; break; } } }}

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:430</a>

### \_onPreDestroy

Called before the object being destroyed.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:463</a>

## \_serialize

The customized serialization for this object. (Editor Only)

meta	description
Returns	<a href="#">object</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:488</a>

Parameters

- `exporting Boolean`

## \_deserialize

Init this object from the custom serialized data.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:498</a>

Parameters

- `data Object` the serialized json data
- `ctx _Deserializer`

# JsonAsset Class

Extends [Asset](#)

Module: [cc](#)

Class for JSON file. When the JSON file is loaded, this object is returned. The parsed JSON object can be accessed through the `json` attribute in it.

If you want to get the original JSON text, you should modify the extname to `.txt` so that it is loaded as a `TextAsset` instead of a `JsonAsset`.

## Index

Properties

- `json Object` The loaded JSON object.

- `loaded` Boolean Whether the asset is loaded or not
- `nativeUrl` String Returns the url of this asset's native object, if none it will returns an empty string.
- `_native` String Serializable url for native asset.
- `_nativeAsset` Object The underlying native asset of this asset if one is available.
- `_uuid` String
- `_name` String
- `_objFlags` Number
- `name` String The name of the object.
- `isValid` Boolean Indicates whether the object is not yet destroyed.

## Methods

- `toString` Returns the asset's url.
- `serialize` 应 AssetDB 要求提供这个方法
- `createNode` Create a new node using this asset in the scene....
- `_setRawAsset` Set native file name for this asset.
- `destroy` Actual object destruction will delayed until before rendering.
- `_destruct` Clear all references in the instance.
- `_onPreDestroy` Called before the object being destroyed.
- `_serialize` The customized serialization for this object.
- `_deserialize` Init this object from the custom serialized data.

## Details

### *Properties*

#### json

The loaded JSON object.

meta	description
Type	<a href="#">Object</a>
Defined in	<a href="#">cocos2d/core/assets/CCJsonAsset.js:47</a>

#### loaded

Whether the asset is loaded or not

meta	description
Type	<a href="#">Boolean</a>

<b>meta</b>	<b>description</b>
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:57</a>

### nativeUrl

Returns the url of this asset's native object, if none it will returns an empty string.

<b>meta</b>	<b>description</b>
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:70</a>

### \_native

Serializable url for native asset.

<b>meta</b>	<b>description</b>
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:108</a>

### \_nativeAsset

The underlying native asset of this asset if one is available. This property can be used to access additional details or functionality related to the asset. This property will be initialized by the loader if \_native is available.

<b>meta</b>	<b>description</b>
Type	<a href="#">Object</a>
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:116</a>

### \_uuid

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/assets/CCRawAsset.js:46</a>

### \_name

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:76</a>

### \_objFlags

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:83</a>

### name

The name of the object.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:243</a>

### Examples

```
obj.name = "New Obj";
```

## isValid

Indicates whether the object is not yet destroyed. (It will not be available after being destroyed)  
When an object's `destroy` is called, it is actually destroyed after the end of this frame.  
So `isValid` will return false from the next frame, while `isValid` in the current frame will still be true.  
If you want to determine whether the current frame has called `destroy`, use `cc.isValid(obj, true)`, but this is often caused by a particular logical requirements, which is not normally required.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:261</a>

## Examples

```
var node = new cc.Node();
cc.log(node.isValid);      // true
node.destroy();
cc.log(node.isValid);      // true, still valid in this frame
// after a frame...
cc.log(node.isValid);      // false, destroyed in the end of last frame
```

## Methods

### toString

Returns the asset's url.

The Asset object overrides the `toString()` method of the Object object. For Asset objects, the `toString()` method returns a string representation of the object. JavaScript calls the `toString()` method automatically when an asset is to be represented as a text value or when a texture is referred to in a string concatenation.

meta	description
Returns	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:165</a>

### serialize

应 AssetDB 要求提供这个方法

meta	description
Returns	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:179</a>

### createNode

Create a new node using this asset in the scene.

If this type of asset dont have its corresponding node type, this method should be null.

meta	description
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:190</a>

### Parameters

- `callback` [Function](#)
- `error` [String](#) null or the error info
- `node` [Object](#) the created node or null

### \_setRawAsset

Set native file name for this asset.

meta	description
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:205</a>

### Parameters

- `filename` [String](#)
- `inLibrary` [Boolean](#)

### destroy

Destroy this Object, and release all its own references to other objects.

Actual object destruction will delayed until before rendering. From the next frame, this object is not usable any more. You can use cc.isValid(obj) to check whether the object is destroyed before accessing it.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:296</a>

## Examples

```
obj.destroy();
```

### \_destruct

Clear all references in the instance.

NOTE: this method will not clear the getter or setter functions which defined in the instance of CCObject. You can override the \_destruct method if you need, for example: \_destruct: function () { for (var key in this) { if (this.hasOwnProperty(key)) { switch (typeof this[key]) { case 'string': this[key] = ''; break; case 'object': case 'function': this[key] = null; break; } } }

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:430</a>

### \_onPreDestroy

Called before the object being destroyed.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:463</a>

### \_serialize

The customized serialization for this object. (Editor Only)

meta	description
Returns	<a href="#">object</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:488</a>

## Parameters

- `exporting Boolean`

`_deserialize`

Init this object from the custom serialized data.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:498</a>

## Parameters

- `data Object` the serialized json data
- `ctx _Deserializer`

# Label Class

Extends [Component](#)

Module: [cc](#)

The Label Component.

## Index

### Properties

- `string Content` Content string of label.
- `horizontalAlign Label.HorizontalAlign` Horizontal Alignment of label.
- `verticalAlign Label.VerticalAlign` Vertical Alignment of label.
- `actualFontSize Number` The actual rendering font size in shrink mode
- `fontSize Number` Font size of label.
- `fontFamily String` Font family of label, only take effect when useSystemFont property is true.
- `lineHeight Number` Line Height of label.
- `overflow Label.Overflow` Overflow of label.
- `enableWrapText Boolean` Whether auto wrap label when string width is large than label width.
- `font Font` The font of label.
- `isSystemFontUsed Boolean` Whether use system font name or not.
- `__eventTargets Array` Register all related EventTargets,...
- `node Node` The node this component is attached to.
- `uuid String` The uid for editor.
- `_enabled Boolean`
- `enabled Boolean` indicates whether this component is enabled or not.

- `enabledInHierarchy` Boolean indicates whether this component is enabled and its node is also active in the hierarchy.
- `_isOnLoadCalled` Number Returns a value which used to indicate the onLoad get called or not.
- `_name` String
- `_objFlags` Number
- `name` String The name of the object.
- `isValid` Boolean Indicates whether the object is not yet destroyed.

## Methods

- `update` This is a lifecycle method.
- `lateUpdate` This is a lifecycle method.
- `__preload` \_\_preload is called before every onLoad.
- `onLoad` When attaching to an active node or its node first activated.
- `start` Called before all scripts' update if the Component is enabled the first time.
- `onEnable` This is a lifecycle method.
- `onDisable` This is a lifecycle method.
- `onDestroy` This is a lifecycle method.
- `onFocusInEditor`
- `onLostFocusInEditor`
- `resetInEditor` Called to initialize the component or node's properties when adding the component the first time or when the Reset command is used.
- `addComponent` Adds a component class to the node.
- `getComponent` Returns the component of supplied type if the node has one attached, null if it doesn't....
- `getComponents` Returns all components of supplied Type in the node.
- `getComponentInChildren` Returns the component of supplied type in any of its children using depth first search.
- `getComponentsInChildren` Returns the components of supplied type in self or any of its children using depth first search.
- `_getLocalBounds` If the component's bounding box is different from the node's, you can implement this method to supply
- `onRestore` for undo/redo operation.
- `schedule` Schedules a custom selector....
- `scheduleOnce` Schedules a callback function that runs only once, with a delay of 0 or larger.
- `unschedule` Unschedules a custom callback function.
- `unscheduleAllCallbacks` unschedule all scheduled callback functions: custom callback functions, and the 'update' callback function....
- `destroy` Actual object destruction will delayed until before rendering.
- `_destruct` Clear all references in the instance.
- `_onPreDestroy` Called before the object being destroyed.
- `_serialize` The customized serialization for this object.
- `_deserialize` Init this object from the custom serialized data.

## Details

## Properties

string

Content string of label.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/components/CCLabel.js:163</a>

horizontalAlign

Horizontal Alignment of label.

meta	description
Type	<a href="#">Label.HorizontalAlign</a>
Defined in	<a href="#">cocos2d/core/components/CCLabel.js:190</a>

verticalAlign

Vertical Alignment of label.

meta	description
Type	<a href="#">Label.VerticalAlign</a>
Defined in	<a href="#">cocos2d/core/components/CCLabel.js:206</a>

actualFontSize

The actual rendering font size in shrink mode

meta	description
Type	<a href="#">Number</a>

meta	description
Defined in	<a href="#">cocos2d/core/components/CCLabel.js:223</a>

### fontSize

Font size of label.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/CCLabel.js:238</a>

### fontFamily

Font family of label, only take effect when useSystemFont property is true.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/components/CCLabel.js:256</a>

### lineHeight

Line Height of label.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/CCLabel.js:272</a>

### overflow

Overflow of label.

meta	description
Type	<a href="#">Label.Overflow</a>
Defined in	<a href="#">cocos2d/core/components/CCLabel.js:288</a>

### enableWrapText

Whether auto wrap label when string width is large than label width.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCLabel.js:305</a>

### font

The font of label.

meta	description
Type	<a href="#">Font</a>
Defined in	<a href="#">cocos2d/core/components/CCLabel.js:327</a>

### isSystemFontUsed

Whether use system font name or not.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCLabel.js:377</a>

## \_\_eventTargets

Register all related EventTargets, all event callbacks will be removed in \_onPreDestroy

meta	description
Type	<a href="#">Array</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:61</a>

## node

The node this component is attached to. A component is always attached to a node.

meta	description
Type	<a href="#">Node</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:75</a>

## Examples

```
cc.log(comp.node);
```

## uuid

The uuid for editor.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:106</a>

## Examples

```
cc.log(comp.uuid);
```

### \_enabled

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:147</a>

### **enabled**

indicates whether this component is enabled or not.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:154</a>

### Examples

```
comp.enabled = true;  
cc.log(comp.enabled);
```

### **enabledInHierarchy**

indicates whether this component is enabled and its node is also active in the hierarchy.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:185</a>

### Examples

```
cc.log(comp.enabledInHierarchy);
```

### \_isOnLoadCalled

Returns a value which used to indicate the onLoad get called or not.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:201</a>

## Examples

```
cc.log(this.\_isOnLoadCalled > 0);
```

`_name`

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:76</a>

`_objFlags`

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:83</a>

`name`

The name of the object.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:243</a>

## Examples

```
obj.name = "New Obj";
```

### isValid

Indicates whether the object is not yet destroyed. (It will not be available after being destroyed)  
When an object's `destroy` is called, it is actually destroyed after the end of this frame.  
So `isValid` will return false from the next frame, while `isValid` in the current frame will still be true.  
If you want to determine whether the current frame has called `destroy`, use `cc.isValid(obj, true)`, but this is often caused by a particular logical requirements, which is not normally required.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:261</a>

## Examples

```
var node = new cc.Node();
cc.log(node.isValid);      // true
node.destroy();
cc.log(node.isValid);      // true, still valid in this frame
// after a frame...
cc.log(node.isValid);      // false, destroyed in the end of last frame
```

## Methods

### update

Update is called every frame, if the Component is enabled.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:222</a>

### Parameters

- `dt` [Number](#) the delta time in seconds it took to complete the last frame

## lateUpdate

LateUpdate is called every frame, if the Component is enabled.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:233</a>

## \_\_preload

\_\_preload is called before every onLoad. It is used to initialize the builtin components internally, to avoid checking whether onLoad is called before every public method calls. This method should be removed if script priority is supported.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:243</a>

## onLoad

When attaching to an active node or its node first activated. onLoad is always called before any start functions, this allows you to order initialization of scripts.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:254</a>

## start

Called before all scripts' update if the Component is enabled the first time. Usually used to initialize some logic which need to be called after all components' onload methods called.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:267</a>

## onEnable

Called when this component becomes enabled and its node is active.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:280</a>

## onDisable

Called when this component becomes disabled or its node becomes inactive.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:290</a>

## onDestroy

Called when this component will be destroyed.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:300</a>

## onFocusInEditor

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:310</a>

## onLostFocusInEditor

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:315</a>

## resetInEditor

Called to initialize the component or node's properties when adding the component the first time or when the Reset command is used. This function is only called in editor.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:320</a>

## addComponent

Adds a component class to the node. You can also add component to node by passing in the name of the script.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:330</a>

### Parameters

- typeOrClassName [Function](#) | [String](#) the constructor or the class name of the component to add

### Examples

```
var sprite = node.addComponent(cc.Sprite);
var test = node.addComponent("Test");
```

## getComponent

Returns the component of supplied type if the node has one attached, null if it doesn't. You can also get component in the node by passing in the name of the script.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:348</a>

#### Parameters

- typeOrClassName [Function](#) | [String](#)

#### Examples

```
// get sprite component.
var sprite = node.getComponent(cc.Sprite);
// get custom test calss.
var test = node.getComponent("Test");
```

### getComponents

Returns all components of supplied Type in the node.

meta	description
Returns	<a href="#">Component[]</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:372</a>

#### Parameters

- typeOrClassName [Function](#) | [String](#)

#### Examples

```
var sprites = node.getComponents(cc.Sprite);
var tests = node.getComponents("Test");
```

### getComponentInChildren

Returns the component of supplied type in any of its children using depth first search.

meta	description
Returns	<a href="#">Component</a>

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:390</a>

## Parameters

- typeOrClassName [Function](#) | [String](#)

## Examples

```
var sprite = node.getComponentInChildren(cc.Sprite);
var Test = node.getComponentInChildren("Test");
```

## getComponentsInChildren

Returns the components of supplied type in self or any of its children using depth first search.

meta	description
Returns	<a href="#">Component[]</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:408</a>

## Parameters

- typeOrClassName [Function](#) | [String](#)

## Examples

```
var sprites = node.getComponentsInChildren(cc.Sprite);
var tests = node.getComponentsInChildren("Test");
```

## \_getLocalBounds

If the component's bounding box is different from the node's, you can implement this method to supply a custom axis aligned bounding box (AABB), so the editor's scene view can perform hit test properly.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:428</a>

## Parameters

- `out_rect Rect` the Rect to receive the bounding box

### onRestore

onRestore is called after the user clicks the Reset item in the Inspector's context menu or performs an undo operation on this component.

If the component contains the "internal state", short for "temporary member variables which not included in its CCClass properties", then you may need to implement this function.

The editor will call the getset accessors of your component to record/restore the component's state for undo/redo operation. However, in extreme cases, it may not work well. Then you should implement this function to manually synchronize your component's "internal states" with its public properties. Once you implement this function, all the getset accessors of your component will not be called when the user performs an undo/redo operation. Which means that only the properties with default value will be recorded or restored by editor.

Similarly, the editor may fail to reset your component correctly in extreme cases. Then if you need to support the reset menu, you should manually synchronize your component's "internal states" with its properties in this function. Once you implement this function, all the getset accessors of your component will not be called during reset operation. Which means that only the properties with default value will be reset by editor.

This function is only called in editor mode.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:441</a>

### schedule

Schedules a custom selector.

If the selector is already scheduled, then the interval parameter will be updated without scheduling it again.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:535</a>

## Parameters

- `callback` [function](#) The callback function
- `interval` [Number](#) Tick interval in seconds. 0 means tick every frame.
- `repeat` [Number](#) The selector will be executed (repeat + 1) times, you can use cc.macro.REPEAT\_FOREVER for tick infinitely.
- `delay` [Number](#) The amount of time that the first tick will wait before execution.

## Examples

```
var timeCallback = function (dt) {
    cc.log("time: " + dt);
}
this.schedule(timeCallback, 1);
```

## scheduleOnce

Schedules a callback function that runs only once, with a delay of 0 or larger.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:572</a>

## Parameters

- `callback` [function](#) A function wrapped as a selector
- `delay` [Number](#) The amount of time that the first tick will wait before execution.

## Examples

```
var timeCallback = function (dt) {
    cc.log("time: " + dt);
}
this.scheduleOnce(timeCallback, 2);
```

## unschedule

Unschedules a custom callback function.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:589</a>

## Parameters

- `callback_fn` [function](#) A function wrapped as a selector

## Examples

```
this.unschedule(_callback);
```

### unscheduleAllCallbacks

unschedule all scheduled callback functions: custom callback functions, and the 'update' callback function.

Actions are not affected by this method.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:605</a>

## Examples

```
this.unscheduleAllCallbacks();
```

### destroy

Destroy this Object, and release all its own references to other objects.

Actual object destruction will delayed until before rendering. From the next frame, this object is not usable any more. You can use `cc.isValid(obj)` to check whether the object is destroyed before accessing it.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:296</a>

## Examples

```
obj.destroy();
```

### \_destruct

Clear all references in the instance.

NOTE: this method will not clear the getter or setter functions which defined in the instance of CCObject. You can override the \_destruct method if you need, for example: \_destruct: function () { for (var key in this) { if (this.hasOwnProperty(key)) { switch (typeof this[key]) { case 'string': this[key] = ""; break; case 'object': case 'function': this[key] = null; break; } } }

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:430</a>

### \_onPreDestroy

Called before the object being destroyed.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:463</a>

### \_serialize

The customized serialization for this object. (Editor Only)

meta	description
Returns	<a href="#">object</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:488</a>

### Parameters

- exporting [Boolean](#)

### \_deserialize

Init this object from the custom serialized data.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:498</a>

## Parameters

- `data Object` the serialized json data
- `ctx _Deserializer`

## LabelAtlas Class

Extends [BitmapFont](#)

Module: [cc](#)

Class for LabelAtlas handling.

## Index

### Properties

- `loaded Boolean` Whether the asset is loaded or not
- `nativeUrl String` Returns the url of this asset's native object, if none it will returns an empty string.
- `_native String` Serializable url for native asset.
- `_nativeAsset Object` The underlying native asset of this asset if one is available.
- `_uuid String`
- `_name String`
- `_objFlags Number`
- `name String` The name of the object.
- `isValid Boolean` Indicates whether the object is not yet destroyed.

### Methods

- `toString` Returns the asset's url.
- `serialize` 应 AssetDB 要求提供这个方法
- `createNode` Create a new node using this asset in the scene....
- `_setRawAsset` Set native file name for this asset.
- `destroy` Actual object destruction will delayed until before rendering.
- `_destruct` Clear all references in the instance.
- `_onPreDestroy` Called before the object being destroyed.
- `_serialize` The customized serialization for this object.
- `_deserialize` Init this object from the custom serialized data.

## Details

## Properties

### loaded

Whether the asset is loaded or not

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:57</a>

### nativeUrl

Returns the url of this asset's native object, if none it will returns an empty string.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:70</a>

### \_native

Serializable url for native asset.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:108</a>

### \_nativeAsset

The underlying native asset of this asset if one is available. This property can be used to access additional details or functionality related to the asset. This property will be initialized by the loader if `_native` is available.

meta	description
Type	<a href="#">Object</a>
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:116</a>

\_uuid

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/assets/CCRawAsset.js:46</a>

\_name

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:76</a>

\_objFlags

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:83</a>

name

The name of the object.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:243</a>

## Examples

```
obj.name = "New Obj";
```

### isValid

Indicates whether the object is not yet destroyed. (It will not be available after being destroyed)  
When an object's `destroy` is called, it is actually destroyed after the end of this frame.  
So `isValid` will return false from the next frame, while `isValid` in the current frame will still be true.  
If you want to determine whether the current frame has called `destroy`, use `cc.isValid(obj, true)`, but this is often caused by a particular logical requirements, which is not normally required.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:261</a>

## Examples

```
var node = new cc.Node();
cc.log(node.isValid);      // true
node.destroy();
cc.log(node.isValid);      // true, still valid in this frame
// after a frame...
cc.log(node.isValid);      // false, destroyed in the end of last frame
```

## Methods

### toString

Returns the asset's url.

The Asset object overrides the `toString()` method of the Object object. For Asset objects, the `toString()` method returns a string representation of the object. JavaScript calls the `toString()` method automatically when an asset is to be represented as a text value or when a texture is referred to in a string concatenation.

meta	description
Returns	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:165</a>

serialize

应 AssetDB 要求提供这个方法

meta	description
Returns	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:179</a>

createNode

Create a new node using this asset in the scene.

If this type of asset dont have its corresponding node type, this method should be null.

meta	description
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:190</a>

Parameters

- callback [Function](#)
- error [String](#) null or the error info
- node [Object](#) the created node or null

\_setRawAsset

Set native file name for this asset.

meta	description
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:205</a>

## Parameters

- `filename` [String](#)
- `inLibrary` [Boolean](#)

## destroy

Destroy this Object, and release all its own references to other objects.

Actual object destruction will delayed until before rendering. From the next frame, this object is not usable any more. You can use `cc.isValid(obj)` to check whether the object is destroyed before accessing it.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:296</a>

## Examples

```
obj.destroy();
```

```
_destruct
```

Clear all references in the instance.

NOTE: this method will not clear the getter or setter functions which defined in the instance of CCObject. You can override the `_destruct` method if you need, for example:

```
_destruct: function () { for (var key in this) { if (this.hasOwnProperty(key)) { switch (typeof this[key]) { case 'string': this[key] = ""; break; case 'object': case 'function': this[key] = null; break; } } }}
```

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:430</a>

```
_onPreDestroy
```

Called before the object being destroyed.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:463</a>

## \_serialize

The customized serialization for this object. (Editor Only)

meta	description
Returns	<a href="#">object</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:488</a>

Parameters

- `exporting Boolean`

## \_deserialize

Init this object from the custom serialized data.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:498</a>

Parameters

- `data Object` the serialized json data
- `ctx _Deserializer`

# LabelOutline Class

Extends [Component](#)

Module: [cc](#)

Outline effect used to change the display, only used for TTF font

## Index

Properties

- `color Color` Change the outline color
- `width Number` Change the outline width
- `__eventTargets Array` Register all related EventTargets,...
- `node Node` The node this component is attached to.

- `uuid` String The uuid for editor.
- `_enabled` Boolean
- `enabled` Boolean indicates whether this component is enabled or not.
- `enabledInHierarchy` Boolean indicates whether this component is enabled and its node is also active in the hierarchy.
- `_isOnLoadCalled` Number Returns a value which used to indicate the onLoad get called or not.
- `_name` String
- `_objFlags` Number
- `name` String The name of the object.
- `isValid` Boolean Indicates whether the object is not yet destroyed.

## Methods

- `update` This is a lifecycle method.
- `lateUpdate` This is a lifecycle method.
- `__preload` \_\_preload is called before every onLoad.
- `onLoad` When attaching to an active node or its node first activated.
- `start` Called before all scripts' update if the Component is enabled the first time.
- `onEnable` This is a lifecycle method.
- `onDisable` This is a lifecycle method.
- `onDestroy` This is a lifecycle method.
- `onFocusInEditor`
- `onLostFocusInEditor`
- `resetInEditor` Called to initialize the component or node's properties when adding the component the first time or when the Reset command is used.
- `addComponent` Adds a component class to the node.
- `getComponent` Returns the component of supplied type if the node has one attached, null if it doesn't....
- `getComponents` Returns all components of supplied Type in the node.
- `getComponentInChildren` Returns the component of supplied type in any of its children using depth first search.
- `getComponentsInChildren` Returns the components of supplied type in self or any of its children using depth first search.
- `_getLocalBounds` If the component's bounding box is different from the node's, you can implement this method to supply
- `onRestore` for undo/redo operation.
- `schedule` Schedules a custom selector....
- `scheduleOnce` Schedules a callback function that runs only once, with a delay of 0 or larger.
- `unschedule` Unschedules a custom callback function.
- `unscheduleAllCallbacks` unschedule all scheduled callback functions: custom callback functions, and the 'update' callback function....
- `destroy` Actual object destruction will delayed until before rendering.
- `_destruct` Clear all references in the instance.
- `_onPreDestroy` Called before the object being destroyed.
- `_serialize` The customized serialization for this object.
- `_deserialize` Init this object from the custom serialized data.

## Details

## Properties

### color

Change the outline color

meta	description
Type	<a href="#">Color</a>
Defined in	<a href="#">cocos2d/core/components/CCLabelOutline.js:56</a>

### Examples

```
outline.color = new cc.Color(0.5, 0.3, 0.7, 1.0);;
```

### width

Change the outline width

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/CCLabelOutline.js:74</a>

### Examples

```
outline.width = 3;
```

### \_\_eventTargets

Register all related EventTargets, all event callbacks will be removed in `_onPreDestroy`

meta	description
Type	<a href="#">Array</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:61</a>

## node

The node this component is attached to. A component is always attached to a node.

meta	description
Type	<a href="#">Node</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:75</a>

## Examples

```
cc.log(comp.node);
```

## uuid

The uuid for editor.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:106</a>

## Examples

```
cc.log(comp.uuid);
```

## \_enabled

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:147</a>

## enabled

indicates whether this component is enabled or not.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:154</a>

## Examples

```
comp.enabled = true;
cc.log(comp.enabled);
```

## enabledInHierarchy

indicates whether this component is enabled and its node is also active in the hierarchy.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:185</a>

## Examples

```
cc.log(comp.enabledInHierarchy);
```

## \_isOnLoadCalled

Returns a value which used to indicate the onLoad get called or not.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:201</a>

## Examples

```
cc.log(this._isOnLoadCalled > 0);
```

### `_name`

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:76</a>

### `_objFlags`

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:83</a>

### `name`

The name of the object.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:243</a>

### Examples

```
obj.name = "New Obj";
```

### `isValid`

Indicates whether the object is not yet destroyed. (It will not be available after being destroyed)  
When an object's `destroy` is called, it is actually destroyed after the end of this frame.  
So `isValid` will return false from the next frame, while `isValid` in the current frame will still be true.  
If you want to determine whether the current frame has called `destroy`, use `cc.isValid(obj, true)`, but this is often caused by a particular logical requirements, which is not normally required.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:261</a>

## Examples

```
var node = new cc.Node();
cc.log(node.isValid);      // true
node.destroy();
cc.log(node.isValid);      // true, still valid in this frame
// after a frame...
cc.log(node.isValid);      // false, destroyed in the end of last frame
```

## Methods

### update

Update is called every frame, if the Component is enabled.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:222</a>

### Parameters

- `dt` [Number](#) the delta time in seconds it took to complete the last frame

### lateUpdate

LateUpdate is called every frame, if the Component is enabled.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:233</a>

## \_\_preload

\_\_preload is called before every onLoad. It is used to initialize the builtin components internally, to avoid checking whether onLoad is called before every public method calls. This method should be removed if script priority is supported.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:243</a>

## onLoad

When attaching to an active node or its node first activated. onLoad is always called before any start functions, this allows you to order initialization of scripts.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:254</a>

## start

Called before all scripts' update if the Component is enabled the first time. Usually used to initialize some logic which need to be called after all components' onload methods called.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:267</a>

## onEnable

Called when this component becomes enabled and its node is active.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:280</a>

### onDisable

Called when this component becomes disabled or its node becomes inactive.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:290</a>

### onDestroy

Called when this component will be destroyed.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:300</a>

### onFocusInEditor

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:310</a>

### onLostFocusInEditor

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:315</a>

### resetInEditor

Called to initialize the component or node's properties when adding the component the first time or when the Reset command is used. This function is only called in editor.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:320</a>

### addComponent

Adds a component class to the node. You can also add component to node by passing in the name of the script.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:330</a>

#### Parameters

- typeOrClassName [Function](#) | [String](#) the constructor or the class name of the component to add

#### Examples

```
var sprite = node.addComponent(cc.Sprite);
var test = node.addComponent("Test");
```

### getComponent

Returns the component of supplied type if the node has one attached, null if it doesn't. You can also get component in the node by passing in the name of the script.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:348</a>

#### Parameters

- typeOrClassName [Function](#) | [String](#)

#### Examples

```
// get sprite component.
```

```
var sprite = node.getComponent(cc.Sprite);
// get custom test class.
var test = node.getComponent("Test");
```

## getComponents

Returns all components of supplied Type in the node.

meta	description
Returns	<a href="#">Component[]</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:372</a>

### Parameters

- typeOrClassName [Function](#) | [String](#)

## Examples

```
var sprites = node.getComponents(cc.Sprite);
var tests = node.getComponents("Test");
```

## getComponentInChildren

Returns the component of supplied type in any of its children using depth first search.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:390</a>

### Parameters

- typeOrClassName [Function](#) | [String](#)

## Examples

```
var sprite = node.getComponentInChildren(cc.Sprite);
var Test = node.getComponentInChildren("Test");
```

## getComponentsInChildren

Returns the components of supplied type in self or any of its children using depth first search.

meta	description
Returns	<a href="#">Component[]</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:408</a>

## Parameters

- typeOrClassName [Function](#) | [String](#)

## Examples

```
var sprites = node.getComponentsInChildren(cc.Sprite);
var tests = node.getComponentsInChildren("Test");
```

## \_getLocalBounds

If the component's bounding box is different from the node's, you can implement this method to supply a custom axis aligned bounding box (AABB), so the editor's scene view can perform hit test properly.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:428</a>

## Parameters

- out\_rect [Rect](#) the Rect to receive the bounding box

## onRestore

onRestore is called after the user clicks the Reset item in the Inspector's context menu or performs an undo operation on this component.

If the component contains the "internal state", short for "temporary member variables which not included in its CCClass properties", then you may need to implement this function.

The editor will call the getset accessors of your component to record/restore the component's state for undo/redo operation. However, in extreme cases, it may not work well. Then you should implement this function to manually synchronize your component's "internal states" with its public properties. Once you implement this function, all the getset accessors of your component will not be called when the user performs an undo/redo operation. Which means that only the properties with default

value  
will be recorded or restored by editor.

Similarly, the editor may failed to reset your component correctly in extreme cases. Then if you need to support the reset menu, you should manually synchronize your component's "internal states" with its properties in this function. Once you implement this function, all the getset accessors of your component will not be called during reset operation. Which means that only the properties with default value will be reset by editor.

This function is only called in editor mode.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:441</a>

## schedule

Schedules a custom selector.

If the selector is already scheduled, then the interval parameter will be updated without scheduling it again.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:535</a>

## Parameters

- `callback function` The callback function
- `interval Number` Tick interval in seconds. 0 means tick every frame.
- `repeat Number` The selector will be executed (repeat + 1) times, you can use `cc.macro.REPEAT_FOREVER` for tick infinitely.
- `delay Number` The amount of time that the first tick will wait before execution.

## Examples

```
var timeCallback = function (dt) {
    cc.log("time: " + dt);
}
this.schedule(timeCallback, 1);
```

## scheduleOnce

Schedules a callback function that runs only once, with a delay of 0 or larger.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:572</a>

## Parameters

- `callback` [function](#) A function wrapped as a selector
- `delay` [Number](#) The amount of time that the first tick will wait before execution.

## Examples

```
var timeCallback = function (dt) {
    cc.log("time: " + dt);
}
this.scheduleOnce(timeCallback, 2);
```

## unschedule

Unschedules a custom callback function.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:589</a>

## Parameters

- `callback_fn` [function](#) A function wrapped as a selector

## Examples

```
this.unschedule(_callback);
```

## unscheduleAllCallbacks

unschedule all scheduled callback functions: custom callback functions, and the 'update' callback function.

Actions are not affected by this method.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:605</a>

## Examples

```
this.unscheduleAllCallbacks();
```

### destroy

Destroy this Object, and release all its own references to other objects.

Actual object destruction will delayed until before rendering. From the next frame, this object is not usable any more. You can use cc.isValid(obj) to check whether the object is destroyed before accessing it.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:296</a>

## Examples

```
obj.destroy();
```

### \_destruct

Clear all references in the instance.

NOTE: this method will not clear the getter or setter functions which defined in the instance of CCObject. You can override the \_destruct method if you need, for example: \_destruct: function () { for (var key in this) { if (this.hasOwnProperty(key)) { switch (typeof this[key]) { case 'string': this[key] = ''; break; case 'object': case 'function': this[key] = null; break; } } }}

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:430</a>

### \_onPreDestroy

Called before the object being destroyed.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:463</a>

## \_serialize

The customized serialization for this object. (Editor Only)

meta	description
Returns	<a href="#">object</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:488</a>

Parameters

- exporting [Boolean](#)

## \_deserialize

Init this object from the custom serialized data.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:498</a>

Parameters

- data [Object](#) the serialized json data
- ctx [\\_Deserializer](#)

# Layout Class

Extends [Component](#)

Module: [cc](#)

The Layout is a container component, use it to arrange child elements easily.

Note :

1. Scaling and rotation of child nodes are not considered.
2. After setting the Layout, the results need to be updated until the next frame, unless you manually call [updateLayout](#).

# Index

## Properties

- `type` Layout.Type The layout type.
- `resizeMode` Layout.ResizeMode The are three resize modes for Layout.
- `cellSize` Size The cell size for grid layout.
- `startAxis` Layout.AxisDirection The start axis for grid layout.
- `paddingLeft` Number The left padding of layout, it only effect the layout in one direction.
- `paddingRight` Number The right padding of layout, it only effect the layout in one direction.
- `paddingTop` Number The top padding of layout, it only effect the layout in one direction.
- `paddingBottom` Number The bottom padding of layout, it only effect the layout in one direction.
- `spacingX` Number The distance in x-axis between each element in layout.
- `spacingY` Number The distance in y-axis between each element in layout.
- `verticalDirection` Layout.VerticalDirection Only take effect in Vertical layout mode.
- `horizontalDirection` Layout.HorizontalDirection Only take effect in Horizontal layout mode.
- `padding` Number The padding of layout, it effects the layout in four direction.
- `__eventTargets` Array Register all related EventTargets,...
- `node` Node The node this component is attached to.
- `uuid` String The uuid for editor.
- `_enabled` Boolean
- `enabled` Boolean indicates whether this component is enabled or not.
- `enabledInHierarchy` Boolean indicates whether this component is enabled and its node is also active in the hierarchy.
- `_isOnLoadCalled` Number Returns a value which used to indicate the onLoad get called or not.
- `_name` String
- `_objFlags` Number
- `name` String The name of the object.
- `isValid` Boolean Indicates whether the object is not yet destroyed.

## Methods

- `updateLayout` Perform the layout update
- `update` This is a lifecycle method.
- `lateUpdate` This is a lifecycle method.
- `__preload` \_\_preload is called before every onLoad.
- `onLoad` When attaching to an active node or its node first activated.
- `start` Called before all scripts' update if the Component is enabled the first time.
- `onEnable` This is a lifecycle method.
- `onDisable` This is a lifecycle method.
- `onDestroy` This is a lifecycle method.
- `onFocusInEditor`
- `onLostFocusInEditor`
- `resetInEditor` Called to initialize the component or node's properties when adding the component the first time or when the Reset command is used.
- `addComponent` Adds a component class to the node.
- `getComponent` Returns the component of supplied type if the node has one attached, null if it doesn't....
- `getComponents` Returns all components of supplied Type in the node.
- `getComponentInChildren` Returns the component of supplied type in any of its children using depth first search.
- `getComponentsInChildren` Returns the components of supplied type in self or any of its children using depth first search.

- `_getLocalBounds` If the component's bounding box is different from the node's, you can implement this method to supply
- `onRestore` for undo/redo operation.
- `schedule` Schedules a custom selector....
- `scheduleOnce` Schedules a callback function that runs only once, with a delay of 0 or larger.
- `unschedule` Unschedules a custom callback function.
- `unscheduleAllCallbacks` unschedule all scheduled callback functions: custom callback functions, and the 'update' callback function....
- `destroy` Actual object destruction will delayed until before rendering.
- `_destruct` Clear all references in the instance.
- `_onPreDestroy` Called before the object being destroyed.
- `_serialize` The customized serialization for this object.
- `_deserialize` Init this object from the custom serialized data.

## Details

### *Properties*

#### type

The layout type.

meta	description
Type	<a href="#">Layout.Type</a>
Defined in	<a href="#">cocos2d/core/components/CCLayout.js:191</a>

#### resizeMode

The are three resize modes for Layout. None, resize Container and resize children.

meta	description
Type	<a href="#">Layout.ResizeMode</a>
Defined in	<a href="#">cocos2d/core/components/CCLayout.js:218</a>

#### cellSize

The cell size for grid layout.

meta	description
Type	<a href="#">Size</a>
Defined in	<a href="#">cocos2d/core/components/CCLayout.js:249</a>

### startAxis

The start axis for grid layout. If you choose horizontal, then children will layout horizontally at first, and then break line on demand. Choose vertical if you want to layout vertically at first .

meta	description
Type	<a href="#">Layout.AxisDirection</a>
Defined in	<a href="#">cocos2d/core/components/CCLayout.js:264</a>

### paddingLeft

The left padding of layout, it only effect the layout in one direction.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/CCLayout.js:290</a>

### paddingRight

The right padding of layout, it only effect the layout in one direction.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/CCLayout.js:303</a>

## paddingTop

The top padding of layout, it only effect the layout in one direction.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/CCLayout.js:316</a>

## paddingBottom

The bottom padding of layout, it only effect the layout in one direction.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/CCLayout.js:329</a>

## spacingX

The distance in x-axis between each element in layout.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/CCLayout.js:342</a>

## spacingY

The distance in y-axis between each element in layout.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/CCLayout.js:355</a>

### **verticalDirection**

Only take effect in Vertical layout mode. This option changes the start element's positioning.

<b>meta</b>	<b>description</b>
Type	<a href="#">Layout.VerticalDirection</a>
Defined in	<a href="#">cocos2d/core/components/CCLayout.js:368</a>

### **horizontalDirection**

Only take effect in Horizontal layout mode. This option changes the start element's positioning.

<b>meta</b>	<b>description</b>
Type	<a href="#">Layout.HorizontalDirection</a>
Defined in	<a href="#">cocos2d/core/components/CCLayout.js:385</a>

### **padding**

The padding of layout, it effects the layout in four direction.

<b>meta</b>	<b>description</b>
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/CCLayout.js:983</a>

### **\_\_eventTargets**

Register all related EventTargets, all event callbacks will be removed in `_onPreDestroy`

<b>meta</b>	<b>description</b>
Type	<a href="#">Array</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:61</a>

## node

The node this component is attached to. A component is always attached to a node.

meta	description
Type	<a href="#">Node</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:75</a>

## Examples

```
cc.log(comp.node);
```

## uuid

The uuid for editor.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:106</a>

## Examples

```
cc.log(comp.uuid);
```

## \_enabled

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:147</a>

## enabled

indicates whether this component is enabled or not.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:154</a>

## Examples

```
comp.enabled = true;
cc.log(comp.enabled);
```

## enabledInHierarchy

indicates whether this component is enabled and its node is also active in the hierarchy.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:185</a>

## Examples

```
cc.log(comp.enabledInHierarchy);
```

## \_isOnLoadCalled

Returns a value which used to indicate the onLoad get called or not.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:201</a>

## Examples

```
cc.log(this._isOnLoadCalled > 0);
```

### `_name`

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:76</a>

### `_objFlags`

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:83</a>

### `name`

The name of the object.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:243</a>

### Examples

```
obj.name = "New Obj";
```

### `isValid`

Indicates whether the object is not yet destroyed. (It will not be available after being destroyed)  
When an object's `destroy` is called, it is actually destroyed after the end of this frame.  
So `isValid` will return false from the next frame, while `isValid` in the current frame will still be true.  
If you want to determine whether the current frame has called `destroy`, use `cc.isValid(obj, true)`, but this is often caused by a particular logical requirements, which is not normally required.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:261</a>

## Examples

```
var node = new cc.Node();
cc.log(node.isValid); // true
node.destroy();
cc.log(node.isValid); // true, still valid in this frame
// after a frame...
cc.log(node.isValid); // false, destroyed in the end of last frame
```

## Methods

### updateLayout

Perform the layout update

meta	description
Defined in	<a href="#">cocos2d/core/components/CCLayout.js:961</a>

## Examples

```
layout.type = cc.Layout.HORIZONTAL;
layout.node.addChild(childNode);
cc.log(childNode.x); // not yet changed
layout.updateLayout();
cc.log(childNode.x); // changed
```

### update

Update is called every frame, if the Component is enabled.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:222</a>

## Parameters

- `dt` Number the delta time in seconds it took to complete the last frame

## lateUpdate

LateUpdate is called every frame, if the Component is enabled.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:233</a>

## \_\_preload

`__preload` is called before every `onLoad`. It is used to initialize the builtin components internally, to avoid checking whether `onLoad` is called before every public method calls. This method should be removed if script priority is supported.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:243</a>

## onLoad

When attaching to an active node or its node first activated. `onLoad` is always called before any start functions, this allows you to order initialization of scripts.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:254</a>

## start

Called before all scripts' update if the Component is enabled the first time. Usually used to initialize some logic which need to be called after all components' onload methods called.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:267</a>

#### onEnable

Called when this component becomes enabled and its node is active.  
 This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:280</a>

#### onDisable

Called when this component becomes disabled or its node becomes inactive.  
 This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:290</a>

#### onDestroy

Called when this component will be destroyed.  
 This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:300</a>

#### onFocusInEditor

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:310</a>

## onLostFocusInEditor

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:315</a>

## resetInEditor

Called to initialize the component or node's properties when adding the component the first time or when the Reset command is used. This function is only called in editor.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:320</a>

## addComponent

Adds a component class to the node. You can also add component to node by passing in the name of the script.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:330</a>

### Parameters

- typeOrClassName [Function](#) | [String](#) the constructor or the class name of the component to add

### Examples

```
var sprite = node.addComponent(cc.Sprite);
var test = node.addComponent("Test");
```

## getComponent

Returns the component of supplied type if the node has one attached, null if it doesn't. You can also get component in the node by passing in the name of the script.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:348</a>

#### Parameters

- typeOrClassName [Function](#) | [String](#)

#### Examples

```
// get sprite component.
var sprite = node.getComponent(cc.Sprite);
// get custom test calss.
var test = node.getComponent("Test");
```

### getComponents

Returns all components of supplied Type in the node.

meta	description
Returns	<a href="#">Component[]</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:372</a>

#### Parameters

- typeOrClassName [Function](#) | [String](#)

#### Examples

```
var sprites = node.getComponents(cc.Sprite);
var tests = node.getComponents("Test");
```

### getComponentInChildren

Returns the component of supplied type in any of its children using depth first search.

meta	description
Returns	<a href="#">Component</a>

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:390</a>

## Parameters

- typeOrClassName [Function](#) | [String](#)

## Examples

```
var sprite = node.getComponentInChildren(cc.Sprite);
var Test = node.getComponentInChildren("Test");
```

## getComponentsInChildren

Returns the components of supplied type in self or any of its children using depth first search.

meta	description
Returns	<a href="#">Component[]</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:408</a>

## Parameters

- typeOrClassName [Function](#) | [String](#)

## Examples

```
var sprites = node.getComponentsInChildren(cc.Sprite);
var tests = node.getComponentsInChildren("Test");
```

## \_getLocalBounds

If the component's bounding box is different from the node's, you can implement this method to supply a custom axis aligned bounding box (AABB), so the editor's scene view can perform hit test properly.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:428</a>

## Parameters

- `out_rect Rect` the Rect to receive the bounding box

### onRestore

onRestore is called after the user clicks the Reset item in the Inspector's context menu or performs an undo operation on this component.

If the component contains the "internal state", short for "temporary member variables which not included in its CCClass properties", then you may need to implement this function.

The editor will call the getset accessors of your component to record/restore the component's state for undo/redo operation. However, in extreme cases, it may not work well. Then you should implement this function to manually synchronize your component's "internal states" with its public properties. Once you implement this function, all the getset accessors of your component will not be called when the user performs an undo/redo operation. Which means that only the properties with default value will be recorded or restored by editor.

Similarly, the editor may fail to reset your component correctly in extreme cases. Then if you need to support the reset menu, you should manually synchronize your component's "internal states" with its properties in this function. Once you implement this function, all the getset accessors of your component will not be called during reset operation. Which means that only the properties with default value will be reset by editor.

This function is only called in editor mode.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:441</a>

### schedule

Schedules a custom selector.

If the selector is already scheduled, then the interval parameter will be updated without scheduling it again.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:535</a>

## Parameters

- `callback function` The callback function
- `interval Number` Tick interval in seconds. 0 means tick every frame.
- `repeat Number` The selector will be executed (repeat + 1) times, you can use cc.macro.REPEAT\_FOREVER for tick infinitely.
- `delay Number` The amount of time that the first tick will wait before execution.

## Examples

```
var timeCallback = function (dt) {
    cc.log("time: " + dt);
}
this.schedule(timeCallback, 1);
```

## scheduleOnce

Schedules a callback function that runs only once, with a delay of 0 or larger.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:572</a>

## Parameters

- `callback function` A function wrapped as a selector
- `delay Number` The amount of time that the first tick will wait before execution.

## Examples

```
var timeCallback = function (dt) {
    cc.log("time: " + dt);
}
this.scheduleOnce(timeCallback, 2);
```

## unschedule

Unschedules a custom callback function.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:589</a>

## Parameters

- `callback_fn` [function](#) A function wrapped as a selector

## Examples

```
this.unschedule(_callback);
```

### unscheduleAllCallbacks

unschedule all scheduled callback functions: custom callback functions, and the 'update' callback function.

Actions are not affected by this method.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:605</a>

## Examples

```
this.unscheduleAllCallbacks();
```

### destroy

Destroy this Object, and release all its own references to other objects.

Actual object destruction will delayed until before rendering. From the next frame, this object is not usable any more. You can use `cc.isValid(obj)` to check whether the object is destroyed before accessing it.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:296</a>

## Examples

```
obj.destroy();
```

### \_destruct

Clear all references in the instance.

NOTE: this method will not clear the getter or setter functions which defined in the instance of CCObject. You can override the \_destruct method if you need, for example: \_destruct: function () { for (var key in this) { if (this.hasOwnProperty(key)) { switch (typeof this[key]) { case 'string': this[key] = ""; break; case 'object': case 'function': this[key] = null; break; } } }

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:430</a>

### \_onPreDestroy

Called before the object being destroyed.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:463</a>

### \_serialize

The customized serialization for this object. (Editor Only)

meta	description
Returns	<a href="#">object</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:488</a>

### Parameters

- exporting [Boolean](#)

### \_deserialize

Init this object from the custom serialized data.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:498</a>

## Parameters

- `data Object` the serialized json data
- `ctx _Deserializer`

## loader Class

Extends [Pipeline](#)

Module: [cc](#)

Loader for resource loading process. It's a singleton object.

## Index

### Properties

- `assetLoader Object` The asset loader in cc.loader's pipeline, it's by default the first pipe.
- `downloader Object` The downloader in cc.loader's pipeline, it's by default the second pipe.
- `loader Object` The loader in cc.loader's pipeline, it's by default the third pipe.

### Methods

- `getXMLHttpRequest` Gets a new XMLHttpRequest instance.
- `addDownloadHandlers` Add custom supported types handler or modify existing type handler for download process.
- `addLoadHandlers` Add custom supported types handler or modify existing type handler for load process.
- `load` Load resources with a progression callback and a complete callback.
- `loadRes` Load resources from the "resources" folder inside the "assets" folder of your project.
- ...
- `loadResArray` This method is like `loadRes` except that it accepts array of url.
- `loadResDir` Load all assets in a folder inside the "assets/resources" folder of your project.
- ...
- `getRes` Get resource data by id.
- `getDependsRecursively` Get all resource dependencies of the requested asset in an array, including itself.
- `release` Release the content of an asset or an array of assets by uuid.
- `releaseAsset` Release the asset by its object.
- `releaseRes` Release the asset loaded by `loadRes`.
- `releaseResDir` Release the all assets loaded by `loadResDir`.
- `releaseAll` Resource all assets.

- `setAutoRelease` according to whether the previous scene checked the "Auto Release Assets" option.
- `setAutoReleaseRecursively` according to whether the previous scene checked the "Auto Release Assets" option.
- `isAutoRelease` Returns whether the asset is configured as auto released, despite how "Auto Release Assets" property is set on scene asset.
- ...
- `constructor` Constructor, pass an array of pipes to construct a new Pipeline,...
- `insertPipe` Insert a new pipe at the given index of the pipeline.
- `insertPipeAfter` Insert a pipe to the end of an existing pipe.
- `appendPipe` Add a new pipe at the end of the pipeline.
- `flowIn` Let new items flow into the pipeline.
- `copyItemStates` Copy the item states from one source item to all destination items.
- `getItem` Returns an item in pipeline.
- `removeItem` Removes an completed item in pipeline.
- `clear` Clear the current pipeline, this function will clean up the items.

## Details

### *Properties*

#### assetLoader

The asset loader in cc.loader's pipeline, it's by default the first pipe. It's used to identify an asset's type, and determine how to download it.

meta	description
Type	<a href="#">Object</a>
Defined in	<a href="#">cocos2d/core/load-pipeline/CCLoader.js:100</a>

#### downloader

The downloader in cc.loader's pipeline, it's by default the second pipe. It's used to download files with several handlers: pure text, image, script, audio, font, uuid. You can add your own download function with addDownloadHandlers

meta	description
Type	<a href="#">Object</a>
Defined in	<a href="#">cocos2d/core/load-pipeline/CCLoader.js:108</a>

## loader

The loader in cc.loader's pipeline, it's by default the third pipe. It's used to parse downloaded content with several handlers: JSON, image, plist, fnt, uuid. You can add your own download function with addLoadHandlers

meta	description
Type	<a href="#">Object</a>
Defined in	<a href="#">cocos2d/core/load-pipeline/CCLoader.js:117</a>

## Methods

### getXMLHttpRequest

Gets a new XMLHttpRequest instance.

meta	description
Returns	XMLHttpRequest
Defined in	<a href="#">cocos2d/core/load-pipeline/CCLoader.js:147</a>

### addDownloadHandlers

Add custom supported types handler or modify existing type handler for download process.

meta	description
Defined in	<a href="#">cocos2d/core/load-pipeline/CCLoader.js:154</a>

## Parameters

- extMap [Object](#) Custom supported types with corresponded handler

## Examples

```
cc.loader.addDownloadHandlers({
    // This will match all url with `scene` extension or all url with `scene` type
    'scene' : function (url, callback) {}
});
```

## addLoadHandlers

Add custom supported types handler or modify existing type handler for load process.

meta	description
Defined in	<a href="#">cocos2d/core/load-pipeline/CCLoader.js:168</a>

### Parameters

- `extMap Object` Custom supported types with corresponded handler

### Examples

```
cc.loader.addLoadHandlers({
    // This will match all url with `scene` extension or all url with `scene` type
    'scene' : function (url, callback) {}
});
```

## load

Load resources with a progression callback and a complete callback. The progression callback is the same as Pipeline's [onProgress](#). The complete callback is almost the same as Pipeline's [onComplete](#). The only difference is when user pass a single url as resources, the complete callback will set its result directly as the second parameter.

meta	description
Defined in	<a href="#">cocos2d/core/load-pipeline/CCLoader.js:182</a>

### Parameters

- `resources String | String[] | Object` Url list in an array
- `progressCallback Function` Callback invoked when progression change
- `completedCount Number` The number of the items that are already completed
- `totalCount Number` The total number of the items
- `item Object` The latest item which flow out the pipeline
- `completeCallback Function` Callback invoked when all resources loaded

### Examples

```
cc.loader.load('a.png', function (err, tex) {
    cc.log('Result should be a texture: ' + (tex instanceof cc.Texture2D));
});

cc.loader.load('http://example.com/a.png', function (err, tex) {
    cc.log('Should load a texture from external url: ' + (tex instanceof cc.Texture2D));
});
```

```

cc.loader.load({url: 'http://example.com/getImageREST?file=a.png', type: 'png'}, function (err,
tex) {
    cc.log('Should load a texture from RESTful API by specify the type: ' + (tex instanceof
cc.Texture2D));
});

cc.loader.load(['a.png', 'b.json'], function (errors, results) {
    if (errors) {
        for (var i = 0; i < errors.length; i++) {
            cc.log('Error url [' + errors[i] + ']: ' + results.getError(errors[i]));
        }
    }
    var aTex = results.getContent('a.png');
    var bJsonObj = results.getContent('b.json');
});

```

## loadRes

Load resources from the "resources" folder inside the "assets" folder of your project.

Note: All asset URLs in Creator use forward slashes, URLs using backslashes will not work.

meta	description
Defined in	<a href="#">cocos2d/core/load-pipeline/CCLoader.js:410</a>

## Parameters

- **url** [String](#) Url of the target resource.  
The url is relative to the "resources" folder, extensions must be omitted.
- **type** [Function](#) Only asset of type will be loaded if this argument is supplied.
- **progressCallback** [Function](#) Callback invoked when progression change.
- **completedCount** [Number](#) The number of the items that are already completed.
- **totalCount** [Number](#) The total number of the items.
- **item** [Object](#) The latest item which flow out the pipeline.
- **completeCallback** [Function](#) Callback invoked when the resource loaded.
- **error** [Error](#) The error info or null if loaded successfully.
- **resource** [Object](#) The loaded resource if it can be found otherwise returns null.

## Examples

```

// load the prefab (project/assets/resources/misc/character/cocos) from resources folder
cc.loader.loadRes('misc/character/cocos', function (err, prefab) {
    if (err) {
        cc.error(err.message || err);
        return;
    }
    cc.log('Result should be a prefab: ' + (prefab instanceof cc.Prefab));
});

// load the sprite frame of (project/assets/resources/imgs/cocos.png) from resources folder
cc.loader.loadRes('imgs/cocos', cc.SpriteFrame, function (err, spriteFrame) {
    if (err) {
        cc.error(err.message || err);
    }
});

```

```

        return;
    }
    cc.log('Result should be a sprite frame: ' + (spriteFrame instanceof cc.SpriteFrame));
});

```

## loadResArray

This method is like [loadRes](#) except that it accepts array of url.

meta	description
Defined in	<a href="#">cocos2d/core/load-pipeline/CCLoader.js:533</a>

### Parameters

- **urls** [String\[\]](#) Array of URLs of the target resource.  
The url is relative to the "resources" folder, extensions must be omitted.
- **type** [Function](#) Only asset of type will be loaded if this argument is supplied.
- **progressCallback** [Function](#) Callback invoked when progression change.
- **completedCount** [Number](#) The number of the items that are already completed.
- **totalCount** [Number](#) The total number of the items.
- **item** [Object](#) The latest item which flow out the pipeline.
- **completeCallback** [Function](#) A callback which is called when all assets have been loaded, or an error occurs.
- **error** [Error](#) If one of the asset failed, the complete callback is immediately called with the error. If all assets are loaded successfully, error will be null.
- **assets** [Asset\[\]](#) | [Array](#) An array of all loaded assets.  
If nothing to load, assets will be an empty array.

### Examples

```
// load the SpriteFrames from resources folder
var spriteFrames;
var urls = ['misc/characters/character_01', 'misc/weapons/weapons_01'];
cc.loader.loadResArray(urls, cc.SpriteFrame, function (err, assets) {
    if (err) {
        cc.error(err);
        return;
    }
    spriteFrames = assets;
    // ...
});
```

## loadResDir

Load all assets in a folder inside the "assets/resources" folder of your project.

Note: All asset URLs in Creator use forward slashes, URLs using backslashes will not work.

meta	description
Defined in	<a href="#">cocos2d/core/load-pipeline/CCLoader.js:591</a>

## Parameters

- **url** [String](#) Url of the target folder.  
The url is relative to the "resources" folder, extensions must be omitted.
- **type** [Function](#) Only asset of type will be loaded if this argument is supplied.
- **progressCallback** [Function](#) Callback invoked when progression change.
- **completedCount** [Number](#) The number of the items that are already completed.
- **totalCount** [Number](#) The total number of the items.
- **item** [Object](#) The latest item which flow out the pipeline.
- **completeCallback** [Function](#) A callback which is called when all assets have been loaded, or an error occurs.
- **error** [Error](#) If one of the asset failed, the complete callback is immediately called with the error. If all assets are loaded successfully, error will be null.
- **assets** [Asset\[\] | Array](#) An array of all loaded assets.  
If nothing to load, assets will be an empty array.
- **urls** [String\[\]](#) An array that lists all the URLs of loaded assets.

## Examples

```
// load the texture (resources/imgs/cocos.png) and the corresponding sprite frame
cc.loader.loadResDir('imgs/cocos', function (err, assets) {
    if (err) {
        cc.error(err);
        return;
    }
    var texture = assets[0];
    var spriteFrame = assets[1];
});

// load all textures in "resources/imgs/"
cc.loader.loadResDir('imgs', cc.Texture2D, function (err, textures) {
    var texture1 = textures[0];
    var texture2 = textures[1];
});

// load all JSONs in "resources/data/"
cc.loader.loadResDir('data', function (err, objects, urls) {
    var data = objects[0];
    var url = urls[0];
});
```

## getRes

Get resource data by id.

When you load resources with [load](#) or [loadRes](#), the url will be the unique identity of the resource. After loaded, you can acquire them by passing the url to this API.

meta	description
Returns	Any
Defined in	<a href="#">cocos2d/core/load-pipeline/CCLoader.js:670</a>

## Parameters

- url [String](#)
- type [Function](#) Only asset of type will be returned if this argument is supplied.

## getDependsRecursively

Get all resource dependencies of the requested asset in an array, including itself. The owner parameter accept the following types: 1. The asset itself; 2. The resource url; 3. The asset's uuid. The returned array stores the dependencies with their uuids, after retrieve dependencies, you can release them, access dependent assets by passing the uuid to [getRes](#), or other stuffs you want. For release all dependencies of an asset, please refer to [release](#). Here is some examples:

meta	description
Returns	<a href="#">Array</a>
Defined in	<a href="#">cocos2d/core/load-pipeline/CCLoader.js:707</a>

## Parameters

- owner [Asset](#) | [RawAsset](#) | [String](#) The owner asset or the resource url or the asset's uuid

## Examples

```
// Release all dependencies of a loaded prefab
var deps = cc.loader.getDependsRecursively(prefab);
cc.loader.release(deps);
// Retrieve all dependent textures
var deps = cc.loader.getDependsRecursively('prefabs/sample');
var textures = [];
for (var i = 0; i < deps.length; ++i) {
    var item = cc.loader.getRes(deps[i]);
    if (item instanceof cc.Texture2D) {
        textures.push(item);
    }
}
```

## release

Release the content of an asset or an array of assets by uuid. Start from v1.3, this method will not only remove the cache of the asset in loader, but also clean up its content. For example, if you release a texture, the texture asset and its gl texture data will be freed up. In complexe project, you can use this function with [getDependsRecursively](#) to free up memory in critical circumstances. Notice, this method may cause the texture to be unusable, if there are still other nodes use the same texture, they may turn to black and report gl errors. If you only want to remove the cache of an asset, please use Pipeline/removeItem:method

meta	description
Defined in	<a href="#">cocos2d/core/load-pipeline/CCLoader.js:748</a>

### Parameters

- **asset** [Asset](#) | [RawAsset](#) | [String](#) | [Array](#)

### Examples

```
// Release a texture which is no longer need
cc.loader.release(texture);
// Release all dependencies of a loaded prefab
var deps = cc.loader.getDependsRecursively('prefabs/sample');
cc.loader.release(deps);
// If there is no instance of this prefab in the scene, the prefab and its dependencies like
textures, sprite frames, etc, will be freed up.
// If you have some other nodes share a texture in this prefab, you can skip it in two ways:
// 1. Forbid auto release a texture before release
cc.loader.setAutoRelease(texture2d, false);
// 2. Remove it from the dependencies array
var deps = cc.loader.getDependsRecursively('prefabs/sample');
var index = deps.indexOf(texture2d._uuid);
if (index !== -1)
    deps.splice(index, 1);
cc.loader.release(deps);
```

## releaseAsset

Release the asset by its object. Refer to [release](#) for detailed informations.

meta	description
Defined in	<a href="#">cocos2d/core/load-pipeline/CCLoader.js:811</a>

### Parameters

- **asset** [Asset](#)

## releaseRes

Release the asset loaded by [loadRes](#). Refer to [release](#) for detailed informations.

meta	description
Defined in	<a href="#">cocos2d/core/load-pipeline/CCLoader.js:825</a>

### Parameters

- url [String](#)
- type [Function](#) Only asset of type will be released if this argument is supplied.

## releaseResDir

Release the all assets loaded by [loadResDir](#). Refer to [release](#) for detailed informations.

meta	description
Defined in	<a href="#">cocos2d/core/load-pipeline/CCLoader.js:843</a>

### Parameters

- url [String](#)
- type [Function](#) Only asset of type will be released if this argument is supplied.

## releaseAll

Resource all assets. Refer to [release](#) for detailed informations.

meta	description
Defined in	<a href="#">cocos2d/core/load-pipeline/CCLoader.js:859</a>

## setAutoRelease

Indicates whether to release the asset when loading a new scene.

By default, when loading a new scene, all assets in the previous scene will be released or preserved according to whether the previous scene checked the "Auto Release Assets" option.

On the other hand, assets dynamically loaded by

using `cc.loader.loadRes` or `cc.loader.loadResDir` will not be affected by that option, remain not released by default.

Use this API to change the default behavior on a single asset, to force preserve or release specified asset when scene switching.

See: [cc.loader.setAutoReleaseRecursively](#), [cc.loader.isAutoRelease](#)

meta	description
Defined in	<a href="#">cocos2d/core/load-pipeline/CCLoader.js:880</a>

## Parameters

- assetOrUrlOrUuid [Asset](#) | [String](#) asset object or the raw asset's url or uuid
- autoRelease [Boolean](#) indicates whether should release automatically

## Examples

```
// auto release the texture event if "Auto Release Assets" disabled in current scene
cc.loader.setAutoRelease(texture2d, true);
// don't release the texture even if "Auto Release Assets" enabled in current scene
cc.loader.setAutoRelease(texture2d, false);
// first parameter can be url
cc.loader.setAutoRelease(audioUrl, false);
```

## setAutoReleaseRecursively

Indicates whether to release the asset and its referenced other assets when loading a new scene. By default, when loading a new scene, all assets in the previous scene will be released or preserved according to whether the previous scene checked the "Auto Release Assets" option. On the other hand, assets dynamically loaded by using `cc.loader.loadRes` or `cc.loader.loadResDir` will not be affected by that option, remain not released by default. Use this API to change the default behavior on the specified asset and its recursively referenced assets, to force preserve or release specified asset when scene switching.

See: [cc.loader.setAutoRelease](#), [cc.loader.isAutoRelease](#)

meta	description
Defined in	<a href="#">cocos2d/core/load-pipeline/CCLoader.js:920</a>

## Parameters

- assetOrUrlOrUuid [Asset](#) | [String](#) asset object or the raw asset's url or uuid
- autoRelease [Boolean](#) indicates whether should release automatically

## Examples

```
// auto release the SpriteFrame and its Texture event if "Auto Release Assets" disabled in
current scene
cc.loader.setAutoReleaseRecursively(spriteFrame, true);
// don't release the SpriteFrame and its Texture even if "Auto Release Assets" enabled in current
scene
```

```
cc.loader.setAutoReleaseRecursively(spriteFrame, false);
// don't release the Prefab and all the referenced assets
cc.loader.setAutoReleaseRecursively(prefab, false);
```

### isAutoRelease

Returns whether the asset is configured as auto released, despite how "Auto Release Assets" property is set on scene asset.

See: [cc.loader.setAutoRelease](#), [cc.loader.setAutoReleaseRecursively](#)

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/load-pipeline/CCLoader.js:968</a>

### Parameters

- `assetOrUrl` [Asset](#) | [String](#) asset object or the raw asset's url

### constructor

Constructor, pass an array of pipes to construct a new Pipeline, the pipes will be chained in the given order.

A pipe is an object which must contain an `id` in string and a `handle` function, the id must be unique in the pipeline.

It can also include `async` property to identify whether it's an asynchronous process.

meta	description
Defined in	<a href="#">cocos2d/core/load-pipeline/pipeline.js:112</a>

### Parameters

- `pipes` [Array](#)

### Examples

```
var pipeline = new Pipeline([
  {
    id: 'Downloader',
    handle: function (item, callback) {},
    async: true
  },
  {id: 'Parser', handle: function (item) {}, async: false}
]);
```

## insertPipe

Insert a new pipe at the given index of the pipeline.

A pipe must contain an `id` in string and a `handle` function, the id must be unique in the pipeline.

meta	description
Defined in	<a href="#">cocos2d/core/load-pipeline/pipeline.js:156</a>

Parameters

- `pipe Object` The pipe to be inserted
- `index Number` The index to insert

## insertPipeAfter

!en Insert a pipe to the end of an existing pipe. The existing pipe must be a valid pipe in the pipeline. !zh 在当前 pipeline 的一个已知 pipe 后面插入一个新的 pipe。

meta	description
Defined in	<a href="#">cocos2d/core/load-pipeline/pipeline.js:199</a>

Parameters

- `refPipe Object` An existing pipe in the pipeline.
- `newPipe Object` The pipe to be inserted.

## appendPipe

Add a new pipe at the end of the pipeline.

A pipe must contain an `id` in string and a `handle` function, the id must be unique in the pipeline.

meta	description
Defined in	<a href="#">cocos2d/core/load-pipeline/pipeline.js:216</a>

Parameters

- `pipe Object` The pipe to be appended

## flowIn

Let new items flow into the pipeline.

Each item can be a simple url string or an object, if it's an object, it must contain `id` property.

You can specify its type by `type` property, by default, the type is the extension name in url.

By adding a `skips` property including pipe ids, you can skip these pipe.

The object can contain any supplementary property as you want.

meta	description
Defined in	<a href="#">cocos2d/core/load-pipeline/pipeline.js:240</a>

### Parameters

- `items` [Array](#)

### Examples

```
pipeline.flowIn([
    'res/Background.png',
    {
        id: 'res/scene.json',
        type: 'scene',
        name: 'scene',
        skips: ['Downloader']
    }
]);
```

## copyItemStates

Copy the item states from one source item to all destination items.

It's quite useful when a pipe generate new items from one source item,

then you should flowIn these generated items into pipeline,

but you probably want them to skip all pipes the source item already go through,

you can achieve it with this API.

For example, an unzip pipe will generate more items, but you won't want them to pass unzip or download pipe again.

meta	description
Defined in	<a href="#">cocos2d/core/load-pipeline/pipeline.js:325</a>

### Parameters

- `srcItem` [Object](#) The source item
- `dstItems` [Array](#) | [Object](#) A single destination item or an array of destination items

## getItem

Returns an item in pipeline.

meta	description
Returns	<a href="#">Object</a>
Defined in	<a href="#">cocos2d/core/load-pipeline/pipeline.js:354</a>

Parameters

- `id Object` The id of the item

## removeItem

Removes an completed item in pipeline. It will only remove the cache in the pipeline or loader, its dependencies won't be released. cc.loader provided another method to completely cleanup the resource and its dependencies, please refer to [cc.loader.release](#)

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/load-pipeline/pipeline.js:374</a>

Parameters

- `id Object` The id of the item

## clear

Clear the current pipeline, this function will clean up the items.

meta	description
Defined in	<a href="#">cocos2d/core/load-pipeline/pipeline.js:394</a>

# LoadingItems Class

Extends [CallbacksInvoker](#)

Module: [cc](#)

LoadingItems is the queue of items which can flow them into the loading pipeline.

Please don't construct it directly, use LoadingItems.create instead, because we use an internal pool to recycle the queues.

It hold a map of items, each entry in the map is a url to object key value pair.

Each item always contains the following property:

- `id`: The identification of the item, usually it's identical to url
- `url`: The url
- `type`: The type, it's the extension name of the url by default, could be specified manually too.
- `error`: The error happened in pipeline will be stored in this property.
- `content`: The content processed by the pipeline, the final result will also be stored in this property.
- `complete`: The flag indicate whether the item is completed by the pipeline.
- `states`: An object stores the states of each pipe the item go through, the state can be:  
`Pipeline.ItemState.WORKING` | `Pipeline.ItemState.ERROR` | `Pipeline.ItemState.COMPLETE`

Item can hold other custom properties.

Each LoadingItems object will be destroyed for recycle after onComplete callback

So please don't hold its reference for later usage, you can copy properties in it though.

## Index

### Properties

- `map` Object The map of all items.
- `completed` Object The map of completed items.
- `totalCount` Number Total count of all items.
- `completedCount` Number Total count of completed items.
- `active` Boolean Activated or not.

### Methods

- `onProgress` This is a callback which will be invoked while an item flow out the pipeline.
- `onComplete` This is a callback which will be invoked while all items is completed,...
- `create` The constructor function of LoadingItems, this will use recycled LoadingItems in the internal pool if possible.
- `getQueue` Retrieve the LoadingItems queue object for an item.
- `itemComplete` Complete an item in the LoadingItems queue, please do not call this method unless you know what's happening.
- `append` Add urls to the LoadingItems queue.
- `allComplete` Complete a LoadingItems queue, please do not call this method unless you know what's happening.
- `isCompleted` Check whether all items are completed.
- `isItemCompleted` Check whether an item is completed.
- `exists` Check whether an item exists.
- `getContent` Returns the content of an internal item.
- `getError` Returns the error of an internal item.
- `addListener` Add a listener for an item, the callback will be invoked when the item is completed.
- `hasListener` Check if the specified key has any registered callback.

- `remove` Removes a listener.
- `removeAllListeners` Removes all callbacks registered in a certain event
- `itemComplete` Complete an item in the LoadingItems queue, please do not call this method unless you know what's happening.
- `destroy` Destroy the LoadingItems queue, the queue object won't be garbage collected, it will be recycled, so every after destroy is not reliable.
- `invoke`
- `add`
- `hasEventListener` Check if the specified key has any registered callback.
- `removeAll` Removes all callbacks registered in a certain event type or all callbacks registered with a certain target

## Details

### *Properties*

#### map

The map of all items.

meta	description
Type	<a href="#">Object</a>
Defined in	<a href="#">cocos2d/core/load-pipeline/loading-items.js:212</a>

#### completed

The map of completed items.

meta	description
Type	<a href="#">Object</a>
Defined in	<a href="#">cocos2d/core/load-pipeline/loading-items.js:220</a>

#### totalCount

Total count of all items.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/load-pipeline/loading-items.js:228</a>

### completedCount

Total count of completed items.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/load-pipeline/loading-items.js:236</a>

### active

Activated or not.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/load-pipeline/loading-items.js:244</a>

## Methods

### onProgress

This is a callback which will be invoked while an item flow out the pipeline. You can pass the callback function in LoadingItems.create or set it later.

meta	description
Defined in	<a href="#">cocos2d/core/load-pipeline/loading-items.js:179</a>

## Parameters

- completedCount [Number](#) The number of the items that are already completed.
- totalCount [Number](#) The total number of the items.
- item [Object](#) The latest item which flow out the pipeline.

## Examples

```
loadingItems.onProgress = function (completedCount, totalCount, item) {
    var progress = (100 * completedCount / totalCount).toFixed(2);
    cc.log(progress + '%');
}
```

## onComplete

This is a callback which will be invoked while all items is completed, You can pass the callback function in LoadingItems.create or set it later.

meta	description
Defined in	<a href="#">cocos2d/core/load-pipeline/loading-items.js:195</a>

## Parameters

- errors [Array](#) All errored urls will be stored in this array, if no error happened, then it will be null
- items [LoadingItems](#) All items.

## Examples

```
loadingItems.onComplete = function (errors, items) {
    if (error)
        cc.log('Completed with ' + errors.length + ' errors');
    else
        cc.log('Completed ' + items.totalCount + ' items');
}
```

## create

The constructor function of LoadingItems, this will use recycled LoadingItems in the internal pool if possible. You can pass onProgress and onComplete callbacks to visualize the loading process.

meta	description
Returns	<a href="#">LoadingItems</a>
Defined in	<a href="#">cocos2d/core/load-pipeline/loading-items.js:292</a>

## Parameters

- `pipeline` [Pipeline](#) The pipeline to process the queue.
- `urlList` [Array](#) The items array.
- `onProgress` [Function](#) The progression callback, refer to `LoadingItems.onProgress`
- `onComplete` [Function](#) The completion callback, refer to `LoadingItems.onComplete`

## Examples

```
cc.LoadingItems.create(cc.loader, ['a.png', 'b.plist'], function (completedCount, totalCount, item) {
    var progress = (100 * completedCount / totalCount).toFixed(2);
    cc.log(progress + '%');
}, function (errors, items) {
    if (errors) {
        for (var i = 0; i < errors.length; ++i) {
            cc.log('Error url: ' + errors[i] + ', error: ' + items.getError(errors[i]));
        }
    } else {
        var result_a = items.getContent('a.png');
        // ...
    }
})
```

## getQueue

Retrieve the `LoadingItems` queue object for an item.

meta	description
Returns	<a href="#">LoadingItems</a>
Defined in	<a href="#">cocos2d/core/load-pipeline/loading-items.js:359</a>

## Parameters

- `item` [Object](#) The item to query

## itemComplete

Complete an item in the `LoadingItems` queue, please do not call this method unless you know what's happening.

meta	description
Defined in	<a href="#">cocos2d/core/load-pipeline/loading-items.js:371</a>

## Parameters

- `item Object` The item which has completed

append

Add urls to the LoadingItems queue.

meta	description
Returns	<a href="#">Array</a>
Defined in	<a href="#">cocos2d/core/load-pipeline/loading-items.js:439</a>

## Parameters

- `urlList Array` The url list to be appended, the url can be object or string

allComplete

Complete a LoadingItems queue, please do not call this method unless you know what's happening.

meta	description
Defined in	<a href="#">cocos2d/core/load-pipeline/loading-items.js:523</a>

isCompleted

Check whether all items are completed.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/load-pipeline/loading-items.js:535</a>

isItemCompleted

Check whether an item is completed.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/load-pipeline/loading-items.js:545</a>

Parameters

- `id` [String](#) The item's id.

exists

Check whether an item exists.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/load-pipeline/loading-items.js:556</a>

Parameters

- `id` [String](#) The item's id.

getContent

Returns the content of an internal item.

meta	description
Returns	<a href="#">Object</a>
Defined in	<a href="#">cocos2d/core/load-pipeline/loading-items.js:567</a>

Parameters

- `id` [String](#) The item's id.

## getError

Returns the error of an internal item.

meta	description
Returns	<a href="#">Object</a>
Defined in	<a href="#">cocos2d/core/load-pipeline/loading-items.js:589</a>

Parameters

- `id` [String](#) The item's id.

## addListener

Add a listener for an item, the callback will be invoked when the item is completed.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/load-pipeline/loading-items.js:610</a>

Parameters

- `key` [String](#)
- `callback` [Function](#) can be null
- `target` [Object](#) can be null

## hasListener

Check if the specified key has any registered callback. If a callback is also specified, it will only return true if the callback is registered.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/load-pipeline/loading-items.js:621</a>

## Parameters

- key [String](#)
- callback [Function](#)
- target [Object](#)

## remove

Removes a listener. It will only remove when key, callback, target all match correctly.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/load-pipeline/loading-items.js:636</a>

## Parameters

- key [String](#)
- callback [Function](#)
- target [Object](#)

## removeAllListeners

Removes all callbacks registered in a certain event type or all callbacks registered with a certain target.

meta	description
Defined in	<a href="#">cocos2d/core/load-pipeline/loading-items.js:651</a>

## Parameters

- key [String](#) | [Object](#) The event key to be removed or the target to be removed

## itemComplete

Complete an item in the LoadingItems queue, please do not call this method unless you know what's happening.

meta	description
Defined in	<a href="#">cocos2d/core/load-pipeline/loading-items.js:683</a>

## Parameters

- `id` [String](#) The item url

## destroy

Destroy the LoadingItems queue, the queue object won't be garbage collected, it will be recycled, so every after destroy is not reliable.

meta	description
Defined in	<a href="#">cocos2d/core/load-pipeline/loading-items.js:723</a>

## invoke

meta	description
Defined in	<a href="#">cocos2d/core/platform/callbacks-invoker.js:236</a>

## Parameters

- `key` [String](#)
- `p1` Any
- `p2` Any
- `p3` Any
- `p4` Any
- `p5` Any

## add

meta	description
Defined in	<a href="#">cocos2d/core/platform/callbacks-invoker.js:97</a>

## Parameters

- `key` [String](#)
- `callback` [Function](#)
- `target` [Object](#) can be null

## hasEventListener

Check if the specified key has any registered callback. If a callback is also specified, it will only return true if the callback is registered.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/callbacks-invoker.js:112</a>

### Parameters

- key [String](#)
- callback [Function](#)
- target [Object](#)

## removeAll

Removes all callbacks registered in a certain event type or all callbacks registered with a certain target

meta	description
Defined in	<a href="#">cocos2d/core/platform/callbacks-invoker.js:154</a>

### Parameters

- keyOrTarget [String](#) | [Object](#) The event key to be removed or the target to be removed

## macro Class

Module: [decorator](#) Parent Module: [cc](#)

Predefined constants

## Index

### Properties

- [RAD](#) Number PI / 180
- [DEG](#) Number One degree

- **REPEAT\_FOREVER** Number
- **FLT\_EPSILON** Number
- **MIN\_ZINDEX** Number Minimum z index value for node
- **MAX\_ZINDEX** Number Maximum z index value for node
- **ONE** Number
- **ZERO** Number
- **SRC\_ALPHA** Number
- **SRC\_ALPHA\_SATURATE** Number
- **SRC\_COLOR** Number
- **DST\_ALPHA** Number
- **DST\_COLOR** Number
- **ONE\_MINUS\_SRC\_ALPHA** Number
- **ONE\_MINUS\_SRC\_COLOR** Number
- **ONE\_MINUS\_DST\_ALPHA** Number
- **ONE\_MINUS\_DST\_COLOR** Number
- **ONE\_MINUS\_CONSTANT\_ALPHA** Number
- **ONE\_MINUS\_CONSTANT\_COLOR** Number
- **ORIENTATION\_PORTRAIT** Number Oriented vertically
- **ORIENTATION\_LANDSCAPE** Number Oriented horizontally
- **ORIENTATION\_AUTO** Number Oriented automatically
- **FIX\_ARTIFACTS\_BY\_STRECHING\_TEXEL\_TMX** Number The same for bottom and top.
- **DIRECTOR\_STATS\_POSITION** Vec2 Position of the FPS (Default: 0,0 (bottom-left corner))...
- **ENABLE\_STACKABLE\_ACTIONS** Number If enabled, actions that alter the position property (eg: CCMoveBy, CCJumpBy, CCBezierBy, etc..) will be stacked.
- **TOUCH\_TIMEOUT** Number The timeout to determine whether a touch is no longer active and should be removed.
- **BATCH\_VERTEX\_COUNT** Number The maximum vertex count for a single batched draw call.
- **ENABLE\_TILEDMAP\_CULLING** Boolean Whether or not enable tiled map auto culling.
- **DOWNLOAD\_MAX\_CONCURRENT** Number The max concurrent task number for the downloader
- **ENABLE\_TRANSPARENT\_CANVAS** Boolean Boolean that indicates if the canvas contains an alpha channel, default sets to false for better performance.
- **ENABLE\_WEBGL\_ANTIALIAS** Boolean Boolean that indicates if the WebGL context is created with antialias option turned on, default value is false.
- **ENABLE\_CULLING** Boolean Whether or not enable auto culling.
- **CLEANUP\_IMAGE\_CACHE** Boolean Whether or not clear dom Image object cache after uploading to gl texture.

## Details

### Properties

RAD

PI / 180

meta	description
Type	<a href="#">Number</a>

<b>meta</b>	<b>description</b>
Defined in	<a href="#">cocos2d/core/platform/CCMacro.js:39</a>

## DEG

One degree

<b>meta</b>	<b>description</b>
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCMacro.js:46</a>

## REPEAT\_FOREVER

<b>meta</b>	<b>description</b>
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCMacro.js:53</a>

## FLT\_EPSILON

<b>meta</b>	<b>description</b>
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCMacro.js:59</a>

## MIN\_ZINDEX

Minimum z index value for node

<b>meta</b>	<b>description</b>
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCMacro.js:65</a>

## MAX\_ZINDEX

Maximum z index value for node

<b>meta</b>	<b>description</b>
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCMacro.js:72</a>

## ONE

<b>meta</b>	<b>description</b>
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCMacro.js:80</a>

## ZERO

<b>meta</b>	<b>description</b>
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCMacro.js:86</a>

## SRC\_ALPHA

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCMacro.js:92</a>

## SRC\_ALPHA\_SATURATE

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCMacro.js:98</a>

## SRC\_COLOR

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCMacro.js:104</a>

## DST\_ALPHA

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCMacro.js:110</a>

## DST\_COLOR

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCMacro.js:116</a>

## ONE\_MINUS\_SRC\_ALPHA

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCMacro.js:122</a>

## ONE\_MINUS\_SRC\_COLOR

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCMacro.js:128</a>

## ONE\_MINUS\_DST\_ALPHA

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCMacro.js:134</a>

## ONE\_MINUS\_DST\_COLOR

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCMacro.js:140</a>

## ONE\_MINUS\_CONSTANT\_ALPHA

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCMacro.js:146</a>

## ONE\_MINUS\_CONSTANT\_COLOR

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCMacro.js:152</a>

## ORIENTATION\_PORTRAIT

Oriented vertically

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCMacro.js:159</a>

## ORIENTATION\_LANDSCAPE

Oriented horizontally

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCMacro.js:166</a>

## ORIENTATION\_AUTO

Oriented automatically

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCMacro.js:173</a>

## FIX\_ARTIFACTS\_BY\_STRECHING\_TEXEL\_TMX

If enabled, the texture coordinates will be calculated by using this formula:

- texCoord.left = (rect.x\*2+1) / (texture.wide\*2);
- texCoord.right = texCoord.left + (rect.width\*2-2)/(texture.wide\*2);

The same for bottom and top.

This formula prevents artifacts by using 99% of the texture.

The "correct" way to prevent artifacts is by expand the texture's border with the same color by 1 pixel

Affected component:

- cc.TMXLayer

Enabled by default. To disabled set it to 0.

To modify it, in Web engine please refer to CCMacro.js, in JSB please refer to CCConfig.h

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCMacro.js:187</a>

## DIRECTOR\_STATS\_POSITION

Position of the FPS (Default: 0,0 (bottom-left corner))

To modify it, in Web engine please refer to CCMacro.js, in JSB please refer to CCConfig.h

meta	description
Type	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/platform/CCMacro.js:209</a>

## ENABLE\_STACKABLE\_ACTIONS

If enabled, actions that alter the position property (eg: CCMoveBy, CCJumpBy, CCBezierBy, etc..) will be stacked.

If you run 2 or more 'position' actions at the same time on a node, then end position will be the sum of all the positions.

If disabled, only the last run action will take effect.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCMacro.js:216</a>

## TOUCH\_TIMEOUT

The timeout to determine whether a touch is no longer active and should be removed. The reason to add this timeout is due to an issue in X5 browser core, when X5 is presented in wechat on Android, if a touch is glissed from the bottom up, and leave the page area, no touch cancel event is triggered, and the touch will be considered active forever. After multiple times of this action, our maximum touches number will be reached and all new touches will be ignored. So this new mechanism can remove the touch that should be inactive if it's not updated during the last 5000 milliseconds. Though it might remove a real touch if it's just not moving for the last 5 seconds which is not easy with the sensibility of mobile touch screen. You can modify this value to have a better behavior if you find it's not enough.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCMacro.js:226</a>

## BATCH\_VERTEX\_COUNT

The maximum vertex count for a single batched draw call.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCMacro.js:248</a>

## ENABLE\_TILEDMAP\_CULLING

Whether or not enabled tiled map auto culling. If you set the TiledMap skew or rotation, then need to manually disable this, otherwise, the rendering will be wrong.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCMacro.js:257</a>

## DOWNLOAD\_MAX\_CONCURRENT

The max concurrent task number for the downloader

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCMacro.js:267</a>

## ENABLE\_TRANSPARENT\_CANVAS

Boolean that indicates if the canvas contains an alpha channel, default sets to false for better performance. Though if you want to make your canvas background transparent and show other dom elements at the background, you can set it to true before `cc.game.run`. Web only.

meta	description
Type	<a href="#">Boolean</a>

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCMacro.js:277</a>

## ENABLE\_WEBGL\_ANTIALIAS

Boolean that indicates if the WebGL context is created with `antialias` option turned on, default value is false. Set it to true could make your game graphics slightly smoother, like texture hard edges when rotated. Whether to use this really depend on your game design and targeted platform, device with retina display usually have good detail on graphics with or without this option, you probably don't want antialias if your game style is pixel art based. Also, it could have great performance impact with some browser / device using software MSAA. You can set it to true before `cc.game.run`. Web only.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCMacro.js:292</a>

## ENABLE\_CULLING

Whether or not enable auto culling. This feature have been removed in v2.0 new renderer due to overall performance consumption. We have no plan currently to re-enable auto culling. If your game have more dynamic objects, we suggest to disable auto culling. If your game have more static objects, we suggest to enable auto culling.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCMacro.js:314</a>
Deprecated	since v2.0

## CLEANUP\_IMAGE\_CACHE

Whether or not clear dom Image object cache after uploading to gl texture. Concretely, we are setting `image.src` to empty string to release the cache. Normally you don't need to enable this option, because on web the Image object doesn't consume too much memory. But on Wechat Game platform, the current version cache decoded data in Image object, which has high memory usage. So we enabled this option by default on Wechat, so that we can release Image cache immediately after uploaded to GPU.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCMacro.js:332</a>

## Manifold Class

Module: [cc](#)

## Index

### Properties

- `type` Number Manifold type : 0: e\_circles, 1: e\_faceA, 2: e\_faceB
- `localPoint` Vec2 -e\_circles: the local center of circleA
- `localNormal` Vec2 -e\_circles: not used
- `points` [ManifoldPoint] the points of contact.

## Details

### *Properties*

#### type

Manifold type : 0: e\_circles, 1: e\_faceA, 2: e\_faceB

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/physics/CCPhysicsContact.js:125</a>

#### localPoint

The local point usage depends on the manifold type: -e\_circles: the local center of circleA - e\_faceA: the center of faceA -e\_faceB: the center of faceB

meta	description
Type	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/physics/CCPhysicsContact.js:134</a>

localNormal

-e\_circles: not used -e\_faceA: the normal on polygonA -e\_faceB: the normal on polygonB

meta	description
Type	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/physics/CCPhysicsContact.js:148</a>

points

the points of contact.

meta	description
Type	<a href="#">[ManifoldPoint]</a>
Defined in	<a href="#">cocos2d/core/physics/CCPhysicsContact.js:161</a>

## ManifoldPoint Class

Module: [cc](#)

A manifold point is a contact point belonging to a contact manifold. It holds details related to the geometry and dynamics of the contact points. Note: the impulses are used for internal caching and may not provide reliable contact forces, especially for high speed collisions.

## Index

### Properties

- `localPoint` Vec2 -e\_circles: the local center of circleB

- `normalImpulse` Number Normal impulse.
- `tangentImpulse` Number Tangent impulse.

## Details

### Properties

#### localPoint

The local point usage depends on the manifold type: -e\_circles: the local center of circleB - e\_faceA: the local center of circleB or the clip point of polygonB -e\_faceB: the clip point of polygonA

meta	description
Type	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/physics/CCPhysicsContact.js:84</a>

#### normalImpulse

Normal impulse.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/physics/CCPhysicsContact.js:97</a>

#### tangentImpulse

Tangent impulse.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/physics/CCPhysicsContact.js:104</a>

## Mask Class

Extends [Component](#)

Module: [cc](#)

The Mask Component

## Index

### Properties

- `type` `Mask.Type` The mask type.
- `spriteFrame` `SpriteFrame` The mask image
- `alphaThreshold` `Number` The content is drawn only where the stencil have pixel with alpha greater than the alphaThreshold.
- `inverted` `Boolean` Reverse mask (Not supported Canvas Mode)
- `segements` `Number` TODO: remove segments, not supported by graphics
- `__eventTargets` `Array` Register all related EventTargets,...
- `node` `Node` The node this component is attached to.
- `uuid` `String` The uuid for editor.
- `_enabled` `Boolean`
- `enabled` `Boolean` indicates whether this component is enabled or not.
- `enabledInHierarchy` `Boolean` indicates whether this component is enabled and its node is also active in the hierarchy.
- `_isOnLoadCalled` `Number` Returns a value which used to indicate the onLoad get called or not.
- `_name` `String`
- `_objFlags` `Number`
- `name` `String` The name of the object.
- `isValid` `Boolean` Indicates whether the object is not yet destroyed.

### Methods

- `update` This is a lifecycle method.
- `lateUpdate` This is a lifecycle method.
- `__preload` \_\_preload is called before every onLoad.
- `onLoad` When attaching to an active node or its node first activated.
- `start` Called before all scripts' update if the Component is enabled the first time.
- `onEnable` This is a lifecycle method.
- `onDisable` This is a lifecycle method.
- `onDestroy` This is a lifecycle method.
- `onFocusInEditor`
- `onLostFocusInEditor`
- `resetInEditor` Called to initialize the component or node's properties when adding the component the first time or when the Reset command is used.
- `addComponent` Adds a component class to the node.
- `getComponent` Returns the component of supplied type if the node has one attached, null if it doesn't....
- `getComponents` Returns all components of supplied Type in the node.
- `getComponentInChildren` Returns the component of supplied type in any of its children using depth first search.
- `getComponentsInChildren` Returns the components of supplied type in self or any of its children using depth first search.

- `_getLocalBounds` If the component's bounding box is different from the node's, you can implement this method to supply
- `onRestore` for undo/redo operation.
- `schedule` Schedules a custom selector....
- `scheduleOnce` Schedules a callback function that runs only once, with a delay of 0 or larger.
- `unschedule` Unschedules a custom callback function.
- `unscheduleAllCallbacks` unschedule all scheduled callback functions: custom callback functions, and the 'update' callback function....
- `destroy` Actual object destruction will delayed until before rendering.
- `_destruct` Clear all references in the instance.
- `_onPreDestroy` Called before the object being destroyed.
- `_serialize` The customized serialization for this object.
- `_deserialize` Init this object from the custom serialized data.

## Details

### *Properties*

type

The mask type.

meta	description
Type	<a href="#">Mask.Type</a>
Defined in	<a href="#">cocos2d/core/components/CCMask.js:92</a>

### Examples

```
mask.type = cc.Mask.Type.RECT;
```

spriteFrame

The mask image

meta	description
Type	<a href="#">SpriteFrame</a>
Defined in	<a href="#">cocos2d/core/components/CCMask.js:122</a>

## Examples

```
mask.spriteFrame = newSpriteFrame;
```

### alphaThreshold

The alpha threshold.(Not supported Canvas Mode)

The content is drawn only where the stencil have pixel with alpha greater than the alphaThreshold.

Should be a float between 0 and 1.

This default to 0 (so alpha test is disabled). When it's set to 1, the stencil will discard all pixels, nothing will be shown, In previous version, it act as if the alpha test is disabled, which is incorrect.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/CCMask.js:154</a>

### inverted

Reverse mask (Not supported Canvas Mode)

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCMask.js:191</a>

### segements

TODO: remove segments, not supported by graphics The segements for ellipse mask.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/CCMask.js:210</a>

### \_\_eventTargets

Register all related EventTargets, all event callbacks will be removed in \_onPreDestroy

<b>meta</b>	<b>description</b>
Type	<a href="#">Array</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:61</a>

## node

The node this component is attached to. A component is always attached to a node.

<b>meta</b>	<b>description</b>
Type	<a href="#">Node</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:75</a>

## Examples

```
cc.log(comp.node);
```

## uuid

The uuid for editor.

<b>meta</b>	<b>description</b>
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:106</a>

## Examples

```
cc.log(comp.uuid);
```

## \_enabled

<b>meta</b>	<b>description</b>
Type	<a href="#">Boolean</a>

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:147</a>

### enabled

indicates whether this component is enabled or not.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:154</a>

### Examples

```
comp.enabled = true;
cc.log(comp.enabled);
```

### enabledInHierarchy

indicates whether this component is enabled and its node is also active in the hierarchy.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:185</a>

### Examples

```
cc.log(comp.enabledInHierarchy);
```

### \_isOnLoadCalled

Returns a value which used to indicate the onLoad get called or not.

meta	description
Type	<a href="#">Number</a>

<b>meta</b>	<b>description</b>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:201</a>

## Examples

```
cc.log(this._isOnLoadCalled > 0);
```

\_name

<b>meta</b>	<b>description</b>
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:76</a>

\_objFlags

<b>meta</b>	<b>description</b>
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:83</a>

name

The name of the object.

<b>meta</b>	<b>description</b>
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:243</a>

## Examples

```
obj.name = "New Obj";
```

## isValid

Indicates whether the object is not yet destroyed. (It will not be available after being destroyed)  
When an object's `destroy` is called, it is actually destroyed after the end of this frame.  
So `isValid` will return false from the next frame, while `isValid` in the current frame will still be true.  
If you want to determine whether the current frame has called `destroy`, use `cc.isValid(obj, true)`, but this is often caused by a particular logical requirements, which is not normally required.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:261</a>

## Examples

```
var node = new cc.Node();
cc.log(node.isValid);      // true
node.destroy();
cc.log(node.isValid);      // true, still valid in this frame
// after a frame...
cc.log(node.isValid);      // false, destroyed in the end of last frame
```

## Methods

### update

Update is called every frame, if the Component is enabled.  
This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:222</a>

### Parameters

- `dt` [Number](#) the delta time in seconds it took to complete the last frame

### lateUpdate

LateUpdate is called every frame, if the Component is enabled.  
This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:233</a>

### \_\_preload

`__preload` is called before every `onLoad`. It is used to initialize the builtin components internally, to avoid checking whether `onLoad` is called before every public method calls. This method should be removed if script priority is supported.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:243</a>

### onLoad

When attaching to an active node or its node first activated. `onLoad` is always called before any start functions, this allows you to order initialization of scripts.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:254</a>

### start

Called before all scripts' update if the Component is enabled the first time. Usually used to initialize some logic which need to be called after all components' `onload` methods called. This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:267</a>

### onEnable

Called when this component becomes enabled and its node is active.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

<b>meta</b>	<b>description</b>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:280</a>

#### onDisable

Called when this component becomes disabled or its node becomes inactive.  
This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

<b>meta</b>	<b>description</b>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:290</a>

#### onDestroy

Called when this component will be destroyed.  
This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

<b>meta</b>	<b>description</b>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:300</a>

#### onFocusInEditor

<b>meta</b>	<b>description</b>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:310</a>

#### onLostFocusInEditor

<b>meta</b>	<b>description</b>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:315</a>

## resetInEditor

Called to initialize the component or node's properties when adding the component the first time or when the Reset command is used. This function is only called in editor.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:320</a>

## addComponent

Adds a component class to the node. You can also add component to node by passing in the name of the script.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:330</a>

### Parameters

- typeOrClassName [Function](#) | [String](#) the constructor or the class name of the component to add

### Examples

```
var sprite = node.addComponent(cc.Sprite);
var test = node.addComponent("Test");
```

## getComponent

Returns the component of supplied type if the node has one attached, null if it doesn't. You can also get component in the node by passing in the name of the script.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:348</a>

## Parameters

- typeOrClassName [Function](#) | [String](#)

## Examples

```
// get sprite component.  
var sprite = node.getComponent(cc.Sprite);  
// get custom test calss.  
var test = node.getComponent("Test");
```

## getComponents

Returns all components of supplied Type in the node.

meta	description
Returns	<a href="#">Component[]</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:372</a>

## Parameters

- typeOrClassName [Function](#) | [String](#)

## Examples

```
var sprites = node.getComponents(cc.Sprite);  
var tests = node.getComponents("Test");
```

## getComponentInChildren

Returns the component of supplied type in any of its children using depth first search.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:390</a>

## Parameters

- typeOrClassName [Function](#) | [String](#)

## Examples

```
var sprite = node.getComponentInChildren(cc.Sprite);
var Test = node.getComponentInChildren("Test");
```

### getComponentsInChildren

Returns the components of supplied type in self or any of its children using depth first search.

meta	description
Returns	<a href="#">Component[]</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:408</a>

## Parameters

- `typeOrClassName` [Function](#) | [String](#)

## Examples

```
var sprites = node.getComponentsInChildren(cc.Sprite);
var tests = node.getComponentsInChildren("Test");
```

### \_getLocalBounds

If the component's bounding box is different from the node's, you can implement this method to supply a custom axis aligned bounding box (AABB), so the editor's scene view can perform hit test properly.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:428</a>

## Parameters

- `out_rect` [Rect](#) the Rect to receive the bounding box

### onRestore

onRestore is called after the user clicks the Reset item in the Inspector's context menu or performs an undo operation on this component.

If the component contains the "internal state", short for "temporary member variables which not included in its CCClass properties", then you may need to implement this function.

The editor will call the getset accessors of your component to record/restore the component's state for undo/redo operation. However, in extreme cases, it may not work well. Then you should implement this function to manually synchronize your component's "internal states" with its public properties. Once you implement this function, all the getset accessors of your component will not be called when the user performs an undo/redo operation. Which means that only the properties with default value will be recorded or restored by editor.

Similarly, the editor may fail to reset your component correctly in extreme cases. Then if you need to support the reset menu, you should manually synchronize your component's "internal states" with its properties in this function. Once you implement this function, all the getset accessors of your component will not be called during reset operation. Which means that only the properties with default value will be reset by editor.

This function is only called in editor mode.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:441</a>

## schedule

Schedules a custom selector.

If the selector is already scheduled, then the interval parameter will be updated without scheduling it again.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:535</a>

## Parameters

- `callback function` The callback function
- `interval Number` Tick interval in seconds. 0 means tick every frame.
- `repeat Number` The selector will be executed (repeat + 1) times, you can use `cc.macro.REPEAT_FOREVER` for tick infinitely.
- `delay Number` The amount of time that the first tick will wait before execution.

## Examples

```
var timeCallback = function (dt) {
    cc.log("time: " + dt);
}
this.schedule(timeCallback, 1);
```

### scheduleOnce

Schedules a callback function that runs only once, with a delay of 0 or larger.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:572</a>

#### Parameters

- `callback` [function](#) A function wrapped as a selector
- `delay` [Number](#) The amount of time that the first tick will wait before execution.

## Examples

```
var timeCallback = function (dt) {
    cc.log("time: " + dt);
}
this.scheduleOnce(timeCallback, 2);
```

### unschedule

Unschedules a custom callback function.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:589</a>

#### Parameters

- `callback_fn` [function](#) A function wrapped as a selector

## Examples

```
this.unschedule(_callback);
```

### unscheduleAllCallbacks

unschedule all scheduled callback functions: custom callback functions, and the 'update' callback function.

Actions are not affected by this method.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:605</a>

## Examples

```
this.unscheduleAllCallbacks();
```

### destroy

Destroy this Object, and release all its own references to other objects.

Actual object destruction will delayed until before rendering. From the next frame, this object is not usable any more. You can use cc.isValid(obj) to check whether the object is destroyed before accessing it.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:296</a>

## Examples

```
obj.destroy();
```

### \_destruct

Clear all references in the instance.

NOTE: this method will not clear the getter or setter functions which defined in the instance of CCObject. You can override the \_destruct method if you need, for example:

```
_destruct: function () { for (var key in this) { if (this.hasOwnProperty(key)) { switch (typeof this[key]) { case 'string': this[key] = ''; break; case 'object': case 'function': this[key] = null; break; } } }}
```

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:430</a>

### \_onPreDestroy

Called before the object being destroyed.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:463</a>

### \_serialize

The customized serialization for this object. (Editor Only)

meta	description
Returns	<a href="#">object</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:488</a>

### Parameters

- `exporting Boolean`

### \_deserialize

Init this object from the custom serialized data.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:498</a>

### Parameters

- `data Object` the serialized json data
- `ctx _Deserializer`

## misc Class

Module: [cc](#)

misc utilities

## Index

## Methods

- [clampf](#) Clamp a value between from and to.
- [clamp01](#) Clamp a value between 0 and 1.
- [lerp](#) Linear interpolation between 2 numbers, the ratio sets how much it is biased to each end
- [degreesToRadians](#) converts degrees to radians
- [radiansToDegrees](#) converts radians to degrees

## Details

### Methods

#### clampf

Clamp a value between from and to.

meta	description
Returns	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/utils/misc.js:157</a>

#### Parameters

- `value` [Number](#)
- `min_inclusive` [Number](#)
- `max_inclusive` [Number](#)

#### Examples

```
var v1 = cc.misc.clampf(20, 0, 20); // 20;
var v2 = cc.misc.clampf(-1, 0, 20); // 0;
var v3 = cc.misc.clampf(10, 0, 20); // 10;
```

#### clamp01

Clamp a value between 0 and 1.

meta	description
Returns	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/utils/misc.js:183</a>

## Parameters

- `value` [Number](#)

## Examples

```
var v1 = cc.misc.clamp01(20); // 1;
var v2 = cc.misc.clamp01(-1); // 0;
var v3 = cc.misc.clamp01(0.5); // 0.5;
```

## lerp

Linear interpolation between 2 numbers, the ratio sets how much it is biased to each end

meta	description
Returns	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/utils/misc.js:198</a>

## Parameters

- `a` [Number](#) number A
- `b` [Number](#) number B
- `r` [Number](#) ratio between 0 and 1

## Examples

```
```Not found for the example path: temp-
src/engine/docs/utils/api/engine/docs/cocos2d/core/platform/CCMacro/lerp.js
```

## degreesToRadians

converts degrees to radians

meta	description
Returns	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/utils/misc.js:211</a>

## Parameters

- `angle` [Number](#)

## radiansToDegrees

converts radians to degrees

meta	description
Returns	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/utils/misc.js:221</a>

Parameters

- `angle` [Number](#)

## MotionStreak Class

Extends [Component](#)

Module: [cc](#)

`cc.MotionStreak` manages a Ribbon based on it's motion in absolute space. You construct it with a fadeTime, minimum segment size, texture path, texture length and color. The fadeTime controls how long it takes each vertex in the streak to fade out, the minimum segment size it how many pixels the streak will move before adding a new ribbon segment, and the texture length is the how many pixels the texture is stretched across. The texture is vertically aligned along the streak segment.

## Index

Properties

- `preview` Boolean
- `fadeTime` Number The fade time to fade.
- `minSeg` Number The minimum segment size.
- `stroke` Number The stroke's width.
- `texture` Texture2D The texture of the MotionStreak.
- `color` Color The color of the MotionStreak.
- `fastMode` Boolean The fast Mode.
- `_eventTargets` Array Register all related EventTargets,...
- `node` Node The node this component is attached to.
- `uuid` String The uuid for editor.
- `_enabled` Boolean
- `enabled` Boolean indicates whether this component is enabled or not.
- `enabledInHierarchy` Boolean indicates whether this component is enabled and its node is also active in the hierarchy.

- `_isOnLoadCalled` Number Returns a value which used to indicate the onLoad get called or not.
- `_name` String
- `_objFlags` Number
- `name` String The name of the object.
- `isValid` Boolean Indicates whether the object is not yet destroyed.

## Methods

- `reset` Remove all living segments of the ribbon.
- `update` This is a lifecycle method.
- `lateUpdate` This is a lifecycle method.
- `__preload` \_\_preload is called before every onLoad.
- `onLoad` When attaching to an active node or its node first activated.
- `start` Called before all scripts' update if the Component is enabled the first time.
- `onEnable` This is a lifecycle method.
- `onDisable` This is a lifecycle method.
- `onDestroy` This is a lifecycle method.
- `onFocusInEditor`
- `onLostFocusInEditor`
- `resetInEditor` Called to initialize the component or node's properties when adding the component the first time or when the Reset command is used.
- `addComponent` Adds a component class to the node.
- `getComponent` Returns the component of supplied type if the node has one attached, null if it doesn't....
- `getComponents` Returns all components of supplied Type in the node.
- `getComponentInChildren` Returns the component of supplied type in any of its children using depth first search.
- `getComponentsInChildren` Returns the components of supplied type in self or any of its children using depth first search.
- `_getLocalBounds` If the component's bounding box is different from the node's, you can implement this method to supply
- `onRestore` for undo/redo operation.
- `schedule` Schedules a custom selector....
- `scheduleOnce` Schedules a callback function that runs only once, with a delay of 0 or larger.
- `unschedule` Unschedules a custom callback function.
- `unscheduleAllCallbacks` unschedule all scheduled callback functions: custom callback functions, and the 'update' callback function....
- `destroy` Actual object destruction will delayed until before rendering.
- `_destruct` Clear all references in the instance.
- `_onPreDestroy` Called before the object being destroyed.
- `_serialize` The customized serialization for this object.
- `_deserialize` Init this object from the custom serialized data.

## Details

## Properties

preview

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCMotionStreak.js:68</a>

fadeTime

The fade time to fade.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/CCMotionStreak.js:83</a>

Examples

```
motionStreak.fadeTime = 3;
```

minSeg

The minimum segment size.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/CCMotionStreak.js:104</a>

Examples

```
motionStreak.minSeg = 3;
```

stroke

The stroke's width.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/CCMotionStreak.js:124</a>

## Examples

```
motionStreak.stroke = 64;
```

### texture

The texture of the MotionStreak.

meta	description
Type	<a href="#">Texture2D</a>
Defined in	<a href="#">cocos2d/core/components/CCMotionStreak.js:144</a>

## Examples

```
motionStreak.texture = newTexture;
```

### color

The color of the MotionStreak.

meta	description
Type	<a href="#">Color</a>
Defined in	<a href="#">cocos2d/core/components/CCMotionStreak.js:179</a>

## Examples

```
motionStreak.color = new cc.Color(255, 255, 255);
```

### fastMode

The fast Mode.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCMotionStreak.js:199</a>

## Examples

```
motionStreak.fastMode = true;
```

### \_\_eventTargets

Register all related EventTargets, all event callbacks will be removed in `_onPreDestroy`

meta	description
Type	<a href="#">Array</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:61</a>

## node

The node this component is attached to. A component is always attached to a node.

meta	description
Type	<a href="#">Node</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:75</a>

## Examples

```
cc.log(comp.node);
```

## uuid

The uuid for editor.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:106</a>

## Examples

```
cc.log(comp.uuid);
```

### \_enabled

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:147</a>

### enabled

indicates whether this component is enabled or not.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:154</a>

## Examples

```
comp.enabled = true;
cc.log(comp.enabled);
```

### enabledInHierarchy

indicates whether this component is enabled and its node is also active in the hierarchy.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:185</a>

## Examples

```
cc.log(comp.enabledInHierarchy);
```

### \_isOnLoadCalled

Returns a value which used to indicate the onLoad get called or not.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:201</a>

## Examples

```
cc.log(this._isOnLoadCalled > 0);
```

### \_name

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:76</a>

### \_objFlags

meta	description
Type	<a href="#">Number</a>

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:83</a>

## name

The name of the object.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:243</a>

## Examples

```
obj.name = "New Obj";
```

## isValid

Indicates whether the object is not yet destroyed. (It will not be available after being destroyed)  
When an object's `destroy` is called, it is actually destroyed after the end of this frame.  
So `isValid` will return false from the next frame, while `isValid` in the current frame will still be true.  
If you want to determine whether the current frame has called `destroy`, use `cc.isValid(obj, true)`, but this is often caused by a particular logical requirements, which is not normally required.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:261</a>

## Examples

```
var node = new cc.Node();
cc.log(node.isValid);    // true
node.destroy();
cc.log(node.isValid);    // true, still valid in this frame
// after a frame...
cc.log(node.isValid);    // false, destroyed in the end of last frame
```

## Methods

### reset

Remove all living segments of the ribbon.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCMotionStreak.js:259</a>

### Examples

```
// Remove all living segments of the ribbon.  
myMotionStreak.reset();
```

### update

Update is called every frame, if the Component is enabled.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:222</a>

### Parameters

- `dt` [Number](#) the delta time in seconds it took to complete the last frame

### lateUpdate

LateUpdate is called every frame, if the Component is enabled.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:233</a>

### \_\_preload

\_\_preload is called before every onLoad. It is used to initialize the builtin components internally, to avoid checking whether onLoad is called before every public method calls. This method should be removed if script priority is supported.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:243</a>

#### onLoad

When attaching to an active node or its node first activated. onLoad is always called before any start functions, this allows you to order initialization of scripts.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:254</a>

#### start

Called before all scripts' update if the Component is enabled the first time. Usually used to initialize some logic which need to be called after all components' onload methods called.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:267</a>

#### onEnable

Called when this component becomes enabled and its node is active.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:280</a>

#### onDisable

Called when this component becomes disabled or its node becomes inactive.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:290</a>

### onDestroy

Called when this component will be destroyed.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:300</a>

### onFocusInEditor

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:310</a>

### onLostFocusInEditor

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:315</a>

### resetInEditor

Called to initialize the component or node's properties when adding the component the first time or when the Reset command is used. This function is only called in editor.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:320</a>

### addComponent

Adds a component class to the node. You can also add component to node by passing in the name of the script.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:330</a>

#### Parameters

- typeOrClassName [Function](#) | [String](#) the constructor or the class name of the component to add

#### Examples

```
var sprite = node.addComponent(cc.Sprite);
var test = node.addComponent("Test");
```

#### getComponent

Returns the component of supplied type if the node has one attached, null if it doesn't. You can also get component in the node by passing in the name of the script.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:348</a>

#### Parameters

- typeOrClassName [Function](#) | [String](#)

#### Examples

```
// get sprite component.
var sprite = node.getComponent(cc.Sprite);
// get custom test calss.
var test = node.getComponent("Test");
```

#### getComponents

Returns all components of supplied Type in the node.

meta	description
Returns	<a href="#">Component[]</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:372</a>

#### Parameters

- typeOrClassName [Function](#) | [String](#)

#### Examples

```
var sprites = node.getComponents(cc.Sprite);
var tests = node.getComponents("Test");
```

### getComponentInChildren

Returns the component of supplied type in any of its children using depth first search.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:390</a>

#### Parameters

- typeOrClassName [Function](#) | [String](#)

#### Examples

```
var sprite = node.getComponentInChildren(cc.Sprite);
var Test = node.getComponentInChildren("Test");
```

### getComponentsInChildren

Returns the components of supplied type in self or any of its children using depth first search.

meta	description
Returns	<a href="#">Component[]</a>

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:408</a>

## Parameters

- `typeOrClassName` [Function](#) | [String](#)

## Examples

```
var sprites = node.getComponentsInChildren(cc.Sprite);
var tests = node.getComponentsInChildren("Test");
```

## \_getLocalBounds

If the component's bounding box is different from the node's, you can implement this method to supply a custom axis aligned bounding box (AABB), so the editor's scene view can perform hit test properly.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:428</a>

## Parameters

- `out_rect` [Rect](#) the Rect to receive the bounding box

## onRestore

onRestore is called after the user clicks the Reset item in the Inspector's context menu or performs an undo operation on this component.

If the component contains the "internal state", short for "temporary member variables which not included in its CCClass properties", then you may need to implement this function.

The editor will call the getset accessors of your component to record/restore the component's state for undo/redo operation. However, in extreme cases, it may not work well. Then you should implement this function to manually synchronize your component's "internal states" with its public properties. Once you implement this function, all the getset accessors of your component will not be called when the user performs an undo/redo operation. Which means that only the properties with default value will be recorded or restored by editor.

Similarly, the editor may failed to reset your component correctly in extreme cases. Then if you need to support the reset menu, you should manually synchronize your component's "internal states" with its properties in this function. Once you implement this function, all the getset accessors of your component will not be called during reset operation. Which means that only the properties with default value will be reset by editor.

This function is only called in editor mode.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:441</a>

### schedule

Schedules a custom selector.

If the selector is already scheduled, then the interval parameter will be updated without scheduling it again.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:535</a>

### Parameters

- `callback` [function](#) The callback function
- `interval` [Number](#) Tick interval in seconds. 0 means tick every frame.
- `repeat` [Number](#) The selector will be executed (`repeat + 1`) times, you can use `cc.macro.REPEAT_FOREVER` for tick infinitely.
- `delay` [Number](#) The amount of time that the first tick will wait before execution.

### Examples

```
var timeCallback = function (dt) {
    cc.log("time: " + dt);
}
this.schedule(timeCallback, 1);
```

### scheduleOnce

Schedules a callback function that runs only once, with a delay of 0 or larger.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:572</a>

## Parameters

- `callback` [function](#) A function wrapped as a selector
- `delay` [Number](#) The amount of time that the first tick will wait before execution.

## Examples

```
var timeCallback = function (dt) {
    cc.log("time: " + dt);
}
this.scheduleOnce(timeCallback, 2);
```

## unschedule

Unschedules a custom callback function.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:589</a>

## Parameters

- `callback_fn` [function](#) A function wrapped as a selector

## Examples

```
this.unschedule(_callback);
```

## unscheduleAllCallbacks

unschedule all scheduled callback functions: custom callback functions, and the 'update' callback function.

Actions are not affected by this method.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:605</a>

## Examples

```
this.unscheduleAllCallbacks();
```

### destroy

Destroy this Object, and release all its own references to other objects.

Actual object destruction will delayed until before rendering. From the next frame, this object is not usable any more. You can use cc.isValid(obj) to check whether the object is destroyed before accessing it.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:296</a>

## Examples

```
obj.destroy();
```

### \_destruct

Clear all references in the instance.

NOTE: this method will not clear the getter or setter functions which defined in the instance of CCObject. You can override the \_destruct method if you need, for example: \_destruct: function () { for (var key in this) { if (this.hasOwnProperty(key)) { switch (typeof this[key]) { case 'string': this[key] = ''; break; case 'object': case 'function': this[key] = null; break; } } }}

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:430</a>

### \_onPreDestroy

Called before the object being destroyed.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:463</a>

## \_serialize

The customized serialization for this object. (Editor Only)

meta	description
Returns	<a href="#">object</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:488</a>

Parameters

- `exporting Boolean`

## \_deserialize

Init this object from the custom serialized data.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:498</a>

Parameters

- `data Object` the serialized json data
- `ctx _Deserializer`

# MotorJoint Class

Extends [Joint](#)

Module: [cc](#) Parent Module: [cc](#)

A motor joint is used to control the relative motion between two bodies. A typical usage is to control the movement of a dynamic body with respect to the ground.

# Index

## Properties

- `anchor Vec2` The anchor of the rigidbody.
- `connectedAnchor Vec2` The anchor of the connected rigidbody.
- `linearOffset Vec2` The linear offset from connected rigidbody to rigidbody.

- `angularOffset` Number The angular offset from connected rigidbody to rigidbody.
- `maxForce` Number The maximum force can be applied to rigidbody.
- `maxTorque` Number The maximum torque can be applied to rigidbody.
- `correctionFactor` Number The position correction factor in the range [0,1].
- `connectedBody` RigidBody The rigidbody to which the other end of the joint is attached.
- `collideConnected` Boolean Should the two rigid bodies connected with this joint collide with each other?
- `__eventTargets` Array Register all related EventTargets,...
- `node` Node The node this component is attached to.
- `uuid` String The uuid for editor.
- `_enabled` Boolean
- `enabled` Boolean indicates whether this component is enabled or not.
- `enabledInHierarchy` Boolean indicates whether this component is enabled and its node is also active in the hierarchy.
- `_isOnLoadCalled` Number Returns a value which used to indicate the onLoad get called or not.
- `_name` String
- `_objFlags` Number
- `name` String The name of the object.
- `isValid` Boolean Indicates whether the object is not yet destroyed.

## Methods

- `apply` Apply current changes to joint, this will regenerate inner box2d joint.
- `getWorldAnchor` Get the anchor point on rigidbody in world coordinates.
- `getWorldConnectedAnchor` Get the anchor point on connected rigidbody in world coordinates.
- `getReactionForce` Gets the reaction force of the joint.
- `getReactionTorque` Gets the reaction torque of the joint.
- `update` This is a lifecycle method.
- `lateUpdate` This is a lifecycle method.
- `__preload` \_\_preload is called before every onLoad.
- `onLoad` When attaching to an active node or its node first activated.
- `start` Called before all scripts' update if the Component is enabled the first time.
- `onEnable` This is a lifecycle method.
- `onDisable` This is a lifecycle method.
- `onDestroy` This is a lifecycle method.
- `onFocusInEditor`
- `onLostFocusInEditor`
- `resetInEditor` Called to initialize the component or node's properties when adding the component the first time or when the Reset command is used.
- `addComponent` Adds a component class to the node.
- `getComponent` Returns the component of supplied type if the node has one attached, null if it doesn't....
- `getComponents` Returns all components of supplied Type in the node.
- `getComponentInChildren` Returns the component of supplied type in any of its children using depth first search.
- `getComponentsInChildren` Returns the components of supplied type in self or any of its children using depth first search.
- `_getLocalBounds` If the component's bounding box is different from the node's, you can implement this method to supply
- `onRestore` for undo/redo operation.
- `schedule` Schedules a custom selector....
- `scheduleOnce` Schedules a callback function that runs only once, with a delay of 0 or larger.

- `unschedule` Unschedules a custom callback function.
- `unscheduleAllCallbacks` unschedule all scheduled callback functions: custom callback functions, and the 'update' callback function....
- `destroy` Actual object destruction will delayed until before rendering.
- `_destruct` Clear all references in the instance.
- `_onPreDestroy` Called before the object being destroyed.
- `_serialize` The customized serialization for this object.
- `_deserialize` Init this object from the custom serialized data.

## Details

### *Properties*

#### anchor

The anchor of the rigidbody.

meta	description
Type	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCMotorJoint.js:60</a>

#### connectedAnchor

The anchor of the connected rigidbody.

meta	description
Type	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCMotorJoint.js:74</a>

#### linearOffset

The linear offset from connected rigidbody to rigidbody.

meta	description
Type	<a href="#">Vec2</a>

<b>meta</b>	<b>description</b>
Defined in	<a href="#">cocos2d/core/physics/joint/CCMotorJoint.js:90</a>

### angularOffset

The angular offset from connected rigidbody to rigidbody.

<b>meta</b>	<b>description</b>
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCMotorJoint.js:111</a>

### maxForce

The maximum force can be applied to rigidbody.

<b>meta</b>	<b>description</b>
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCMotorJoint.js:132</a>

### maxTorque

The maximum torque can be applied to rigidbody.

<b>meta</b>	<b>description</b>
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCMotorJoint.js:153</a>

### correctionFactor

The position correction factor in the range [0,1].

<b>meta</b>	<b>description</b>
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCMotorJoint.js:174</a>

### connectedBody

The rigidbody to which the other end of the joint is attached.

<b>meta</b>	<b>description</b>
Type	<a href="#">RigidBody</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCJoint.js:74</a>

### collideConnected

Should the two rigid bodies connected with this joint collide with each other?

<b>meta</b>	<b>description</b>
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCJoint.js:88</a>

### \_\_eventTargets

Register all related EventTargets, all event callbacks will be removed in \_onPreDestroy

<b>meta</b>	<b>description</b>
Type	<a href="#">Array</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:61</a>

## node

The node this component is attached to. A component is always attached to a node.

meta	description
Type	<a href="#">Node</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:75</a>

## Examples

```
cc.log(comp.node);
```

## uuid

The uuid for editor.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:106</a>

## Examples

```
cc.log(comp.uuid);
```

## \_enabled

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:147</a>

## enabled

indicates whether this component is enabled or not.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:154</a>

## Examples

```
comp.enabled = true;
cc.log(comp.enabled);
```

## enabledInHierarchy

indicates whether this component is enabled and its node is also active in the hierarchy.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:185</a>

## Examples

```
cc.log(comp.enabledInHierarchy);
```

## \_isOnLoadCalled

Returns a value which used to indicate the onLoad get called or not.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:201</a>

## Examples

```
cc.log(this._isOnLoadCalled > 0);
```

### `_name`

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:76</a>

### `_objFlags`

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:83</a>

### `name`

The name of the object.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:243</a>

### Examples

```
obj.name = "New Obj";
```

### `isValid`

Indicates whether the object is not yet destroyed. (It will not be available after being destroyed)  
When an object's `destroy` is called, it is actually destroyed after the end of this frame.  
So `isValid` will return false from the next frame, while `isValid` in the current frame will still be true.  
If you want to determine whether the current frame has called `destroy`, use `cc.isValid(obj, true)`, but this is often caused by a particular logical requirements, which is not normally required.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:261</a>

## Examples

```
var node = new cc.Node();
cc.log(node.isValid);      // true
node.destroy();
cc.log(node.isValid);      // true, still valid in this frame
// after a frame...
cc.log(node.isValid);      // false, destroyed in the end of last frame
```

## Methods

### apply

Apply current changes to joint, this will regenerate inner box2d joint.

meta	description
Defined in	<a href="#">cocos2d/core/physics/joint/CCJoint.js:115</a>

### getWorldAnchor

Get the anchor point on rigidbody in world coordinates.

meta	description
Returns	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCJoint.js:127</a>

### getWorldConnectedAnchor

Get the anchor point on connected rigidbody in world coordinates.

meta	description
Returns	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCJoint.js:143</a>

### getReactionForce

Gets the reaction force of the joint.

meta	description
Returns	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCJoint.js:159</a>

#### Parameters

- `timeStep` [Number](#) The time to calculate the reaction force for.

### getReactionTorque

Gets the reaction torque of the joint.

meta	description
Returns	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCJoint.js:176</a>

#### Parameters

- `timeStep` [Number](#) The time to calculate the reaction torque for.

### update

Update is called every frame, if the Component is enabled.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:222</a>

## Parameters

- `dt Number` the delta time in seconds it took to complete the last frame

## lateUpdate

LateUpdate is called every frame, if the Component is enabled.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:233</a>

## \_\_preload

`__preload` is called before every `onLoad`. It is used to initialize the builtin components internally, to avoid checking whether `onLoad` is called before every public method calls. This method should be removed if script priority is supported.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:243</a>

## onLoad

When attaching to an active node or its node first activated. `onLoad` is always called before any start functions, this allows you to order initialization of scripts.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:254</a>

## start

Called before all scripts' update if the Component is enabled the first time. Usually used to initialize some logic which need to be called after all components' `onload` methods called. This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:267</a>

## onEnable

Called when this component becomes enabled and its node is active.  
This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:280</a>

## onDisable

Called when this component becomes disabled or its node becomes inactive.  
This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:290</a>

## onDestroy

Called when this component will be destroyed.  
This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:300</a>

## onFocusInEditor

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:310</a>

## onLostFocusInEditor

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:315</a>

## resetInEditor

Called to initialize the component or node's properties when adding the component the first time or when the Reset command is used. This function is only called in editor.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:320</a>

## addComponent

Adds a component class to the node. You can also add component to node by passing in the name of the script.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:330</a>

### Parameters

- typeOrClassName [Function](#) | [String](#) the constructor or the class name of the component to add

### Examples

```
var sprite = node.addComponent(cc.Sprite);
var test = node.addComponent("Test");
```

## getComponent

Returns the component of supplied type if the node has one attached, null if it doesn't. You can also get component in the node by passing in the name of the script.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:348</a>

### Parameters

- `typeOrClassName` [Function](#) | [String](#)

### Examples

```
// get sprite component.  
var sprite = node.getComponent(cc.Sprite);  
// get custom test calss.  
var test = node.getComponent("Test");
```

## getComponents

Returns all components of supplied Type in the node.

meta	description
Returns	<a href="#">Component[]</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:372</a>

### Parameters

- `typeOrClassName` [Function](#) | [String](#)

### Examples

```
var sprites = node.getComponents(cc.Sprite);  
var tests = node.getComponents("Test");
```

## getComponentInChildren

Returns the component of supplied type in any of its children using depth first search.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:390</a>

## Parameters

- typeOrClassName [Function](#) | [String](#)

## Examples

```
var sprite = node.getComponentInChildren(cc.Sprite);
var Test = node.getComponentInChildren("Test");
```

## getComponentsInChildren

Returns the components of supplied type in self or any of its children using depth first search.

meta	description
Returns	<a href="#">Component[]</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:408</a>

## Parameters

- typeOrClassName [Function](#) | [String](#)

## Examples

```
var sprites = node.getComponentsInChildren(cc.Sprite);
var tests = node.getComponentsInChildren("Test");
```

## \_getLocalBounds

If the component's bounding box is different from the node's, you can implement this method to supply a custom axis aligned bounding box (AABB), so the editor's scene view can perform hit test properly.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:428</a>

## Parameters

- `out_rect Rect` the Rect to receive the bounding box

### onRestore

onRestore is called after the user clicks the Reset item in the Inspector's context menu or performs an undo operation on this component.

If the component contains the "internal state", short for "temporary member variables which not included in its CCClass properties", then you may need to implement this function.

The editor will call the getset accessors of your component to record/restore the component's state for undo/redo operation. However, in extreme cases, it may not work well. Then you should implement this function to manually synchronize your component's "internal states" with its public properties. Once you implement this function, all the getset accessors of your component will not be called when the user performs an undo/redo operation. Which means that only the properties with default value will be recorded or restored by editor.

Similarly, the editor may fail to reset your component correctly in extreme cases. Then if you need to support the reset menu, you should manually synchronize your component's "internal states" with its properties in this function. Once you implement this function, all the getset accessors of your component will not be called during reset operation. Which means that only the properties with default value will be reset by editor.

This function is only called in editor mode.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:441</a>

## schedule

Schedules a custom selector.

If the selector is already scheduled, then the interval parameter will be updated without scheduling it again.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:535</a>

### Parameters

- `callback` [function](#) The callback function
- `interval` [Number](#) Tick interval in seconds. 0 means tick every frame.
- `repeat` [Number](#) The selector will be executed (repeat + 1) times, you can use cc.macro.REPEAT\_FOREVER for tick infinitely.
- `delay` [Number](#) The amount of time that the first tick will wait before execution.

## Examples

```
var timeCallback = function (dt) {
    cc.log("time: " + dt);
}
this.schedule(timeCallback, 1);
```

## scheduleOnce

Schedules a callback function that runs only once, with a delay of 0 or larger.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:572</a>

### Parameters

- `callback` [function](#) A function wrapped as a selector
- `delay` [Number](#) The amount of time that the first tick will wait before execution.

## Examples

```
var timeCallback = function (dt) {
    cc.log("time: " + dt);
}
this.scheduleOnce(timeCallback, 2);
```

## unschedule

Unschedules a custom callback function.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:589</a>

## Parameters

- `callback_fn` [function](#) A function wrapped as a selector

## Examples

```
this.unschedule(_callback);
```

### unscheduleAllCallbacks

unschedule all scheduled callback functions: custom callback functions, and the 'update' callback function.

Actions are not affected by this method.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:605</a>

## Examples

```
this.unscheduleAllCallbacks();
```

### destroy

Destroy this Object, and release all its own references to other objects.

Actual object destruction will delayed until before rendering. From the next frame, this object is not usable any more. You can use `cc.isValid(obj)` to check whether the object is destroyed before accessing it.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:296</a>

## Examples

```
obj.destroy();
```

### \_destruct

Clear all references in the instance.

NOTE: this method will not clear the getter or setter functions which defined in the instance of CCObject. You can override the \_destruct method if you need, for example: \_destruct: function () { for (var key in this) { if (this.hasOwnProperty(key)) { switch (typeof this[key]) { case 'string': this[key] = ""; break; case 'object': case 'function': this[key] = null; break; } } }

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:430</a>

### \_onPreDestroy

Called before the object being destroyed.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:463</a>

### \_serialize

The customized serialization for this object. (Editor Only)

meta	description
Returns	<a href="#">object</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:488</a>

### Parameters

- exporting [Boolean](#)

### \_deserialize

Init this object from the custom serialized data.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:498</a>

## Parameters

- `data Object` the serialized json data
- `ctx _Deserializer`

## MouseJoint Class

Extends [Joint](#)

Module: [cc](#) Parent Module: [cc](#)

A mouse joint is used to make a point on a body track a specified world point. This a soft constraint with a maximum force. This allows the constraint to stretch and without applying huge forces. Mouse Joint will auto register the touch event with the mouse region node, and move the choosed rigidbody in touch move event. Note : generally mouse joint only used in test bed.

## Index

### Properties

- `anchor Vec2` The anchor of the rigidbody.
- `connectedAnchor Vec2` The anchor of the connected rigidbody.
- `mouseRegion Node` The node used to register touch evnet.
- `target Vec2` The target point.
- `frequency Number` The spring frequency.
- `0 Number` The damping ratio.
- `maxForce Number` The maximum force
- `connectedBody RigidBody` The rigidbody to which the other end of the joint is attached.
- `collideConnected Boolean` Should the two rigid bodies connected with this joint collide with each other?
- `__eventTargets Array` Register all related EventTargets,...
- `node Node` The node this component is attached to.
- `uuid String` The uuid for editor.
- `_enabled Boolean`
- `enabled Boolean` indicates whether this component is enabled or not.
- `enabledInHierarchy Boolean` indicates whether this component is enabled and its node is also active in the hierarchy.
- `_isOnLoadCalled Number` Returns a value which used to indicate the onLoad get called or not.
- `_name String`
- `_objFlags Number`
- `name String` The name of the object.
- `isValid Boolean` Indicates whether the object is not yet destroyed.

## Methods

- `apply` Apply current changes to joint, this will regenerate inner box2d joint.
- `getWorldAnchor` Get the anchor point on rigidbody in world coordinates.
- `getWorldConnectedAnchor` Get the anchor point on connected rigidbody in world coordinates.
- `getReactionForce` Gets the reaction force of the joint.
- `getReactionTorque` Gets the reaction torque of the joint.
- `update` This is a lifecycle method.
- `lateUpdate` This is a lifecycle method.
- `__preload` \_\_preload is called before every onLoad.
- `onLoad` When attaching to an active node or its node first activated.
- `start` Called before all scripts' update if the Component is enabled the first time.
- `onEnable` This is a lifecycle method.
- `onDisable` This is a lifecycle method.
- `onDestroy` This is a lifecycle method.
- `onFocusInEditor`
- `onLostFocusInEditor`
- `resetInEditor` Called to initialize the component or node's properties when adding the component the first time or when the Reset command is used.
- `addComponent` Adds a component class to the node.
- `getComponent` Returns the component of supplied type if the node has one attached, null if it doesn't....
- `getComponents` Returns all components of supplied Type in the node.
- `getComponentInChildren` Returns the component of supplied type in any of its children using depth first search.
- `getComponentsInChildren` Returns the components of supplied type in self or any of its children using depth first search.
- `_getLocalBounds` If the component's bounding box is different from the node's, you can implement this method to supply
- `onRestore` for undo/redo operation.
- `schedule` Schedules a custom selector....
- `scheduleOnce` Schedules a callback function that runs only once, with a delay of 0 or larger.
- `unschedule` Unschedules a custom callback function.
- `unscheduleAllCallbacks` unschedule all scheduled callback functions: custom callback functions, and the 'update' callback function....
- `destroy` Actual object destruction will delayed until before rendering.
- `_destruct` Clear all references in the instance.
- `_onPreDestroy` Called before the object being destroyed.
- `_serialize` The customized serialization for this object.
- `_deserialize` Init this object from the custom serialized data.

## Details

### *Properties*

#### anchor

The anchor of the rigidbody.

meta	description
Type	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCMouseJoint.js:78</a>

#### connectedAnchor

The anchor of the connected rigidbody.

meta	description
Type	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCMouseJoint.js:92</a>

#### mouseRegion

The node used to register touch event. If this is null, it will be the joint's node.

meta	description
Type	<a href="#">Node</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCMouseJoint.js:107</a>

#### target

The target point. The mouse joint will move choosed rigidbody to target point.

meta	description
Type	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCMouseJoint.js:123</a>

## frequency

The spring frequency.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCMouseJoint.js:147</a>

## 0

The damping ratio.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCMouseJoint.js:168</a>

## maxForce

The maximum force

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCMouseJoint.js:189</a>

## connectedBody

The rigidbody to which the other end of the joint is attached.

meta	description
Type	<a href="#">RigidBody</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCJoint.js:74</a>

## collideConnected

Should the two rigid bodies connected with this joint collide with each other?

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCJoint.js:88</a>

## \_\_eventTargets

Register all related EventTargets, all event callbacks will be removed in `_onPreDestroy`

meta	description
Type	<a href="#">Array</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:61</a>

## node

The node this component is attached to. A component is always attached to a node.

meta	description
Type	<a href="#">Node</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:75</a>

## Examples

```
cc.log(comp.node);
```

## uuid

The uuid for editor.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:106</a>

## Examples

```
cc.log(comp.uuid);
```

### \_enabled

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:147</a>

### enabled

indicates whether this component is enabled or not.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:154</a>

## Examples

```
comp.enabled = true;
cc.log(comp.enabled);
```

### enabledInHierarchy

indicates whether this component is enabled and its node is also active in the hierarchy.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:185</a>

## Examples

```
cc.log(comp.enabledInHierarchy);
```

### \_isOnLoadCalled

Returns a value which used to indicate the onLoad get called or not.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:201</a>

## Examples

```
cc.log(this.\_isOnLoadCalled > 0);
```

### \_name

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:76</a>

### \_objFlags

meta	description
Type	<a href="#">Number</a>

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:83</a>

## name

The name of the object.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:243</a>

## Examples

```
obj.name = "New Obj";
```

## isValid

Indicates whether the object is not yet destroyed. (It will not be available after being destroyed)  
When an object's `destroy` is called, it is actually destroyed after the end of this frame.  
So `isValid` will return false from the next frame, while `isValid` in the current frame will still be true.  
If you want to determine whether the current frame has called `destroy`, use `cc.isValid(obj, true)`, but this is often caused by a particular logical requirements, which is not normally required.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:261</a>

## Examples

```
var node = new cc.Node();
cc.log(node.isValid);      // true
node.destroy();
cc.log(node.isValid);    // true, still valid in this frame
// after a frame...
cc.log(node.isValid);    // false, destroyed in the end of last frame
```

## Methods

### apply

Apply current changes to joint, this will regenerate inner box2d joint.

meta	description
Defined in	<a href="#">cocos2d/core/physics/joint/CCJoint.js:115</a>

### getWorldAnchor

Get the anchor point on rigidbody in world coordinates.

meta	description
Returns	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCJoint.js:127</a>

### getWorldConnectedAnchor

Get the anchor point on connected rigidbody in world coordinates.

meta	description
Returns	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCJoint.js:143</a>

### getReactionForce

Gets the reaction force of the joint.

meta	description
Returns	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCJoint.js:159</a>

## Parameters

- `timeStep` [Number](#) The time to calculate the reaction force for.

`getReactionTorque`

Gets the reaction torque of the joint.

meta	description
Returns	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/physics/joint/CCJoint.js:176</a>

## Parameters

- `timeStep` [Number](#) The time to calculate the reaction torque for.

`update`

Update is called every frame, if the Component is enabled.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:222</a>

## Parameters

- `dt` [Number](#) the delta time in seconds it took to complete the last frame

`lateUpdate`

LateUpdate is called every frame, if the Component is enabled.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:233</a>

## \_\_preload

\_\_preload is called before every onLoad. It is used to initialize the builtin components internally, to avoid checking whether onLoad is called before every public method calls. This method should be removed if script priority is supported.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:243</a>

## onLoad

When attaching to an active node or its node first activated. onLoad is always called before any start functions, this allows you to order initialization of scripts.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:254</a>

## start

Called before all scripts' update if the Component is enabled the first time. Usually used to initialize some logic which need to be called after all components' onload methods called.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:267</a>

## onEnable

Called when this component becomes enabled and its node is active.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:280</a>

### onDisable

Called when this component becomes disabled or its node becomes inactive.  
This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:290</a>

### onDestroy

Called when this component will be destroyed.  
This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:300</a>

### onFocusInEditor

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:310</a>

### onLostFocusInEditor

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:315</a>

### resetInEditor

Called to initialize the component or node's properties when adding the component the first time or when the Reset command is used. This function is only called in editor.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:320</a>

### addComponent

Adds a component class to the node. You can also add component to node by passing in the name of the script.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:330</a>

#### Parameters

- typeOrClassName [Function](#) | [String](#) the constructor or the class name of the component to add

#### Examples

```
var sprite = node.addComponent(cc.Sprite);
var test = node.addComponent("Test");
```

### getComponent

Returns the component of supplied type if the node has one attached, null if it doesn't. You can also get component in the node by passing in the name of the script.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:348</a>

#### Parameters

- typeOrClassName [Function](#) | [String](#)

#### Examples

```
// get sprite component.
```

```
var sprite = node.getComponent(cc.Sprite);
// get custom test class.
var test = node.getComponent("Test");
```

## getComponents

Returns all components of supplied Type in the node.

meta	description
Returns	<a href="#">Component[]</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:372</a>

### Parameters

- typeOrClassName [Function](#) | [String](#)

## Examples

```
var sprites = node.getComponents(cc.Sprite);
var tests = node.getComponents("Test");
```

## getComponentInChildren

Returns the component of supplied type in any of its children using depth first search.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:390</a>

### Parameters

- typeOrClassName [Function](#) | [String](#)

## Examples

```
var sprite = node.getComponentInChildren(cc.Sprite);
var Test = node.getComponentInChildren("Test");
```

## getComponentsInChildren

Returns the components of supplied type in self or any of its children using depth first search.

meta	description
Returns	<a href="#">Component[]</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:408</a>

## Parameters

- typeOrClassName [Function](#) | [String](#)

## Examples

```
var sprites = node.getComponentsInChildren(cc.Sprite);
var tests = node.getComponentsInChildren("Test");
```

## \_getLocalBounds

If the component's bounding box is different from the node's, you can implement this method to supply a custom axis aligned bounding box (AABB), so the editor's scene view can perform hit test properly.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:428</a>

## Parameters

- out\_rect [Rect](#) the Rect to receive the bounding box

## onRestore

onRestore is called after the user clicks the Reset item in the Inspector's context menu or performs an undo operation on this component.

If the component contains the "internal state", short for "temporary member variables which not included in its CCClass properties", then you may need to implement this function.

The editor will call the getset accessors of your component to record/restore the component's state for undo/redo operation. However, in extreme cases, it may not work well. Then you should implement this function to manually synchronize your component's "internal states" with its public properties. Once you implement this function, all the getset accessors of your component will not be called when the user performs an undo/redo operation. Which means that only the properties with default

value  
will be recorded or restored by editor.

Similarly, the editor may failed to reset your component correctly in extreme cases. Then if you need to support the reset menu, you should manually synchronize your component's "internal states" with its properties in this function. Once you implement this function, all the getset accessors of your component will not be called during reset operation. Which means that only the properties with default value will be reset by editor.

This function is only called in editor mode.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:441</a>

## schedule

Schedules a custom selector.

If the selector is already scheduled, then the interval parameter will be updated without scheduling it again.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:535</a>

## Parameters

- `callback function` The callback function
- `interval Number` Tick interval in seconds. 0 means tick every frame.
- `repeat Number` The selector will be executed (repeat + 1) times, you can use `cc.macro.REPEAT_FOREVER` for tick infinitely.
- `delay Number` The amount of time that the first tick will wait before execution.

## Examples

```
var timeCallback = function (dt) {
    cc.log("time: " + dt);
}
this.schedule(timeCallback, 1);
```

## scheduleOnce

Schedules a callback function that runs only once, with a delay of 0 or larger.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:572</a>

## Parameters

- `callback` [function](#) A function wrapped as a selector
- `delay` [Number](#) The amount of time that the first tick will wait before execution.

## Examples

```
var timeCallback = function (dt) {
    cc.log("time: " + dt);
}
this.scheduleOnce(timeCallback, 2);
```

## unschedule

Unschedules a custom callback function.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:589</a>

## Parameters

- `callback_fn` [function](#) A function wrapped as a selector

## Examples

```
this.unschedule(_callback);
```

## unscheduleAllCallbacks

unschedule all scheduled callback functions: custom callback functions, and the 'update' callback function.

Actions are not affected by this method.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:605</a>

## Examples

```
this.unscheduleAllCallbacks();
```

### destroy

Destroy this Object, and release all its own references to other objects.

Actual object destruction will delayed until before rendering. From the next frame, this object is not usable any more. You can use cc.isValid(obj) to check whether the object is destroyed before accessing it.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:296</a>

## Examples

```
obj.destroy();
```

### \_destruct

Clear all references in the instance.

NOTE: this method will not clear the getter or setter functions which defined in the instance of CCObject. You can override the \_destruct method if you need, for example: \_destruct: function () { for (var key in this) { if (this.hasOwnProperty(key)) { switch (typeof this[key]) { case 'string': this[key] = ''; break; case 'object': case 'function': this[key] = null; break; } } }}

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:430</a>

### \_onPreDestroy

Called before the object being destroyed.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:463</a>

## \_serialize

The customized serialization for this object. (Editor Only)

meta	description
Returns	<a href="#">object</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:488</a>

Parameters

- exporting [Boolean](#)

## \_deserialize

Init this object from the custom serialized data.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:498</a>

Parameters

- data [Object](#) the serialized json data
- ctx [\\_Deserializer](#)

# Node Class

Extends [\\_BaseNode](#)

Defined in: <https://github.com/cocos-creator/engine/blob/4f734a806d1fd7c4073fb064fddc961384fe67af/cocos2d/core/CCNode.js:510>

Module: [cc](#)

Class of all entities in Cocos Creator scenes.

For events supported by Node, please refer to [Node.EventType](#)

# Index

## Properties

- `groupIndex` Integer Group index of node....
- `group` String Group of node....
- `position` Vec2 The position (x, y) of the node in its parent's coordinates.
- `x` Number X axis position of node.
- `y` Number Y axis position of node.
- `rotation` Number Rotation of node.
- `rotationX` Number Rotation on x axis.
- `rotationY` Number Rotation on y axis.
- `scale` Number The local scale relative to the parent.
- `scaleX` Number Scale on x axis.
- `scaleY` Number Scale on y axis.
- `skewX` Number Skew x
- `skewY` Number Skew y
- `opacity` Number Opacity of node, default value is 255.
- `color` Color Color of node, default value is white: (255, 255, 255).
- `anchorX` Number Anchor point's position on x axis.
- `anchorY` Number Anchor point's position on y axis.
- `width` Number Width of node.
- `height` Number Height of node.
- `zIndex` Number Node's order in children list will affect its rendering order.
- `cascadeOpacity` Boolean Cascade opacity is removed from v2.0
- `_level` Number
- `_components` Component[]
- `_prefab` PrefabInfo The PrefabInfo object
- `_persistNode` Boolean If true, the node is an persist node which won't be destroyed during scene transition.
- `name` String Name of node.
- `uuid` String The uid for editor, will be stripped before building project.
- `children` Node[] All children nodes.
- `childrenCount` Number All children nodes.
- `active` Boolean The local active state of this node....
- `activeInHierarchy` Boolean Indicates whether this node is active in the scene.
- `_eventTargets` EventTarget[] Register all related EventTargets,...
- `parent` Node The parent of the node.
- `_name` String
- `_objFlags` Number
- `isValid` Boolean Indicates whether the object is not yet destroyed.

## Methods

- `constructor`
- `on` 1.
- `once` Register an callback of a specific event type on the Node,...
- `off` Removes the callback previously registered with the same type, callback, target and or useCapture.
- `targetOff` Removes all callbacks previously registered with the same target.

- `hasEventListener` Checks whether the EventTarget object has any callback registered for a specific type of event.
- `emit` Trigger an event directly with the event name and necessary arguments.
- `dispatchEvent` Dispatches an event into the event flow.
- `pauseSystemEvents` Pause node related system events registered with the current Node.
- `resumeSystemEvents` Resume node related system events registered with the current Node.
- `_getCapturingTargets` Get all the targets listening to the supplied type of event in the target's capturing phase.
- `_getBubblingTargets` Get all the targets listening to the supplied type of event in the target's bubbling phase.
- `runAction` The node becomes the action's target.
- `pauseAllActions` Pause all actions running on the current node.
- `resumeAllActions` Resume all paused actions on the current node.
- `stopAllActions` Stops and removes all actions from the running action list .
- `stopAction` Stops and removes an action from the running action list.
- `stopActionByTag` Removes an action from the running action list by its tag.
- `getActionByTag` Returns an action from the running action list by its tag.
- `getNumberOfRunningActions` Composable actions are counted as 1 action.
- `getPosition` Returns a copy of the position (x, y) of the node in its parent's coordinates.
- `setPosition` Sets the position (x, y) of the node in its parent's coordinates....
- `getScale` Returns the scale factor of the node.
- `setScale` Sets the scale factor of the node.
- `setRotation` Set rotation of node (along z axi).
- `getRotation` Get rotation of node (along z axi).
- `getContentSize` Returns a copy the untransformed size of the node.
- `setContentSize` All nodes has a size.
- `getAnchorPoint` It's like a pin in the node where it is "attached" to its parent.
- `setAnchorPoint` Sets the anchor point in percent.
- `lookAt` Set rotation by lookAt target point, normally used by Camera Node
- `getLocalMatrix` Get the local transform matrix (4x4), based on parent node coordinates
- `getWorldMatrix` Get the world transform matrix (4x4)
- `convertToNodeSpace` Converts a Point to node (local) space coordinates then add the anchor point position.
- `convertToWorldSpace` Converts a Point related to the left bottom corner of the node's bounding box to world space coordinates.
- `convertToNodeSpaceAR` Converts a Point to node (local) space coordinates in which the anchor point is the origin position.
- `convertToWorldSpaceAR` Converts a Point in node coordinates to world space coordinates.
- `getNodeToParentTransform` Returns the matrix that transform the node's (local) space coordinates into the parent's space coordinates....
- `getNodeToParentTransformAR` Returns the matrix that transform the node's (local) space coordinates into the parent's space coordinates....
- `getNodeToWorldTransform` Returns the world affine transform matrix.
- `getNodeToWorldTransformAR` Returns the world affine transform matrix.
- `getParentToNodeTransform` The matrix is in Pixels.
- `getWorldToNodeTransform` Returns the inverse world affine transform matrix.
- `convertTouchToNodeSpace` convenience methods which take a cc.Touch instead of cc.Vec2.
- `convertTouchToNodeSpaceAR` converts a cc.Touch (world coordinates) into a local coordinate.
- `getBoundingBox` Returns a "local" axis aligned bounding box of the node.
- `getBoundingBoxToWorld` Returns a "world" axis aligned bounding box of the node....

- `addChild` Adds a child to the node with z order and name.
- `cleanup` Stops all running actions and schedulers.
- `sortAllChildren` Sorts the children array depends on children's zIndex and arrivalOrder,...
- `getDisplayedOpacity` Returns the displayed opacity of Node,...
- `getDisplayedColor` Returns the displayed color of Node,...
- `isCascadeOpacityEnabled` Cascade opacity is removed from v2.0
- `setCascadeOpacityEnabled` Cascade opacity is removed from v2.0
- `setOpacityModifyRGB` Opacity modify RGB have been removed since v2.0
- `isOpacityModifyRGB` Opacity modify RGB have been removed since v2.0
- `attr` Properties configuration function ...
- `getChildByUuid` Returns a child from the container given its uuid.
- `getChildByName` Returns a child from the container given its name.
- `insertChild` Inserts a child to the node at a specified index.
- `getSiblingIndex` Get the sibling index.
- `setSiblingIndex` Set the sibling index of this node.
- `walk` Walk though the sub children tree of the current node.
- `removeFromParent` Remove itself from its parent node.
- `removeChild` Removes a child from the container.
- `removeAllChildren` Removes all children from the container and do a cleanup all running actions depending on the cleanup parameter.
- `isChildOf` Is this node a child of the given node?
- `getComponent` Returns the component of supplied type if the node has one attached, null if it doesn't....
- `getComponents` Returns all components of supplied type in the node.
- `getComponentInChildren` Returns the component of supplied type in any of its children using depth first search.
- `getComponentsInChildren` Returns all components of supplied type in self or any of its children.
- `addComponent` Adds a component class to the node.
- `_addComponentAt` This api should only used by undo system
- `removeComponent` Removes a component identified by the given name or removes the component object given.
- `_getDependComponent`
- `destroyAllChildren` Destroy all children from the node, and release all their own references to other objects....
- `destroy` Actual object destruction will delayed until before rendering.
- `_destruct` Clear all references in the instance.
- `_onPreDestroy` Called before the object being destroyed.
- `_serialize` The customized serialization for this object.
- `_deserialize` Init this object from the custom serialized data.

## Events

- `position-changed`
- `size-changed`
- `anchor-changed`
- `child-added`
- `child-removed`
- `child-reorder`
- `group-changed`

- **active-in-hierarchy-changed** Note: This event is only emitted from the top most node whose active value did changed,...

## Details

### *Properties*

#### groupIndex

Group index of node.

Which Group this node belongs to will resolve that this node's collision components can collide with which other collision components.

meta	description
Type	Integer
Defined in	<a href="#">cocos2d/core/CCNode.js:553</a>

#### group

Group of node.

Which Group this node belongs to will resolve that this node's collision components can collide with which other collision components.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:569</a>

#### position

The position (x, y) of the node in its parent's coordinates.

meta	description
Type	<a href="#">Vec2</a>

meta	description
Defined in	<a href="#">cocos2d/core/CCNode.js:592</a>

## Examples

```
cc.log("Node Position: " + node.position);
```

x

x axis position of node.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:600</a>

## Examples

```
node.x = 100;
cc.log("Node Position X: " + node.x);
```

y

y axis position of node.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:643</a>

## Examples

```
node.y = 100;
cc.log("Node Position Y: " + node.y);
```

rotation

Rotation of node.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:709</a>

## Examples

```
node.rotation = 90;
cc.log("Node Rotation: " + node.rotation);
```

## rotationX

Rotation on x axis.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:737</a>

## Examples

```
node.rotationX = 45;
cc.log("Node Rotation X: " + node.rotationX);
```

## rotationY

Rotation on y axis.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:770</a>

## Examples

```
node.rotationY = 45;
cc.log("Node Rotation Y: " + node.rotationY);
```

## scale

The local scale relative to the parent.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:803</a>

## Examples

```
node.scale = 1;
```

## scaleX

Scale on x axis.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:812</a>

## Examples

```
node.scaleX = 0.5;
cc.log("Node Scale X: " + node.scaleX);
```

## scaleY

Scale on y axis.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:838</a>

## Examples

```
node.scaleY = 0.5;
cc.log("Node Scale Y: " + node.scaleY);
```

## skewX

Skew x

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:864</a>

## Examples

```
node.skewX = 0;
cc.log("Node SkewX: " + node.skewX);
```

## skewY

Skew y

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:884</a>

## Examples

```
node.skewY = 0;
cc.log("Node SkewY: " + node.skewY);
```

## opacity

Opacity of node, default value is 255.

meta	description
Type	<a href="#">Number</a>

meta	description
Defined in	<a href="#">cocos2d/core/CCNode.js:904</a>

## Examples

```
node.opacity = 255;
```

### color

Color of node, default value is white: (255, 255, 255).

meta	description
Type	<a href="#">Color</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:925</a>

## Examples

```
node.color = new cc.Color(255, 255, 255);
```

### anchorX

Anchor point's position on x axis.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:951</a>

## Examples

```
node.anchorX = 0;
```

### anchorY

Anchor point's position on y axis.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:974</a>

## Examples

```
node.anchorY = 0;
```

## width

Width of node.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:997</a>

## Examples

```
node.width = 100;
```

## height

Height of node.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:1027</a>

## Examples

```
node.height = 100;
```

## zIndex

zIndex is the 'key' used to sort the node relative to its siblings.

The value of zIndex should be in the range between cc.macro.MIN\_ZINDEX and cc.macro.MAX\_ZINDEX.

The Node's parent will sort all its children based on the zIndex value and the arrival order.

Nodes with greater zIndex will be sorted after nodes with smaller zIndex.

If two nodes have the same zIndex, then the node that was added first to the children's array will be in front of the other node in the array.

Node's order in children list will affect its rendering order. Parent is always rendering before all children.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:1057</a>

## Examples

```
node.zIndex = 1;  
cc.log("Node zIndex: " + node.zIndex);
```

## cascadeOpacity

Cascade opacity is removed from v2.0 Indicate whether node's opacity value affect its child nodes, default value is true.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:3087</a>
Deprecated	since v2.0

### \_level

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/utils/base-node.js:151</a>

### \_components

meta	description
Type	<a href="#">Component[]</a>
Defined in	<a href="#">cocos2d/core/utils/base-node.js:159</a>

### \_prefab

The PrefabInfo object

meta	description
Type	PrefabInfo
Defined in	<a href="#">cocos2d/core/utils/base-node.js:168</a>

### \_persistNode

If true, the node is an persist node which won't be destroyed during scene transition.  
If false, the node will be destroyed automatically when loading a new scene. Default is false.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/utils/base-node.js:176</a>

## name

Name of node.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/utils/base-node.js:200</a>

## Examples

```
node.name = "New Node";
cc.log("Node Name: " + node.name);
```

## uuid

The uuid for editor, will be stripped before building project.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/utils/base-node.js:222</a>

## Examples

```
cc.log("Node Uuid: " + node.uuid);
```

## children

All children nodes.

meta	description
Type	<a href="#">Node[]</a>
Defined in	<a href="#">cocos2d/core/utils/base-node.js:237</a>

## Examples

```
var children = node.children;
for (var i = 0; i < children.length; ++i) {
    cc.log("Node: " + children[i]);
}
```

### childrenCount

All children nodes.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/utils/base-node.js:255</a>

## Examples

```
var count = node.childrenCount;
cc.log("Node Children Count: " + count);
```

### active

The local active state of this node.

Note that a Node may be inactive because a parent is not active, even if this returns true.

Use Node/activeInHierarchy:property if you want to check if the Node is actually treated as active in the scene.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/utils/base-node.js:271</a>

## Examples

```
node.active = false;
```

### activeInHierarchy

Indicates whether this node is active in the scene.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/utils/base-node.js:305</a>

## Examples

```
cc.log("activeInHierarchy: " + node.activeInHierarchy);
```

### \_\_eventTargets

Register all related EventTargets, all event callbacks will be removed in `_onPreDestroy`

meta	description
Type	<a href="#">EventTarget[]</a>
Defined in	<a href="#">cocos2d/core/utils/base-node.js:331</a>

## parent

The parent of the node.

meta	description
Type	<a href="#">Node</a>
Defined in	<a href="#">cocos2d/core/utils/base-node.js:343</a>

## Examples

```
node.parent = newNode;
```

## \_name

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:76</a>

## \_objFlags

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:83</a>

## isValid

Indicates whether the object is not yet destroyed. (It will not be available after being destroyed)

When an object's `destroy` is called, it is actually destroyed after the end of this frame. So `isValid` will return false from the next frame, while `isValid` in the current frame will still be true. If you want to determine whether the current frame has called `destroy`, use `cc.isValid(obj, true)`, but this is often caused by a particular logical requirements, which is not normally required.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:261</a>

## Examples

```
var node = new cc.Node();
cc.log(node.isValid);    // true
node.destroy();
cc.log(node.isValid);    // true, still valid in this frame
// after a frame...
cc.log(node.isValid);    // false, destroyed in the end of last frame
```

## Methods

constructor

meta	description
Defined in	<a href="#">cocos2d/core/CCNode.js:1100</a>

Parameters

- `name` [String](#)

on

Register a callback of a specific event type on Node.

Use this method to register touch or mouse event permit propagation based on scene graph,

These kinds of event are triggered with dispatchEvent, the dispatch process has three steps:

1. Capturing phase: dispatch in capture targets (`_getCapturingTargets`), e.g. parents in node tree, from root to the real target
2. At target phase: dispatch to the listeners of the real target
3. Bubbling phase: dispatch in bubble targets (`_getBubblingTargets`), e.g. parents in node tree, from the real target to root

In any moment of the dispatching process, it can be stopped via `event.stopPropagation()` or `event.stopPropagationImmediate()`.

It's the recommended way to register touch/mouse event for Node, please do not use `cc.eventManager` directly for Node.

You can also register custom event and use `emit` to trigger custom event on Node.

For such events, there won't be capturing and bubbling phase, your event will be dispatched directly to its listeners registered on the same node.

You can also pass event callback parameters with `emit` by passing parameters after `type`.

meta	description
Returns	<a href="#">Function</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:1398</a>

## Parameters

- type [String](#) | [Node.EventType](#) A string representing the event type to listen for.  
See Node/EventTyupe/POSITION\_CHANGED for all builtin events.
- callback [Function](#) The callback that will be invoked when the event is dispatched. The callback is ignored if it is a duplicate (the callbacks are unique).
- event [Event](#) | Any event or first argument when emit
- arg2 Any arg2
- arg3 Any arg3
- arg4 Any arg4
- arg5 Any arg5
- target [Object](#) The target (this object) to invoke the callback, can be null
- useCapture [Boolean](#) When set to true, the listener will be triggered at capturing phase which is ahead of the final target emit, otherwise it will be triggered during bubbling phase.

## Examples

```
this.node.on(cc.Node.EventType.TOUCH_START, this.memberFunction, this); // if  
"this" is component and the "memberFunction" declared in CCClass.  
node.on(cc.Node.EventType.TOUCH_START, callback, this);  
node.on(cc.Node.EventType.TOUCH_MOVE, callback, this);  
node.on(cc.Node.EventType.TOUCH_END, callback, this);  
node.on(cc.Node.EventType.TOUCH_CANCEL, callback, this);  
node.on(cc.Node.EventType.ANCHOR_CHANGED, callback);
```

once

Register an callback of a specific event type on the Node, the callback will remove itself after the first time it is triggered.

meta	description
Defined in	<a href="#">cocos2d/core/CCNode.js:1473</a>

## Parameters

- type [String](#) A string representing the event type to listen for.
- callback [Function](#) The callback that will be invoked when the event is dispatched.
  - The callback is ignored if it is a duplicate (the callbacks are unique).
- event [Event](#) | Any event or first argument when emit
- arg2 Any arg2
- arg3 Any arg3
- arg4 Any arg4
- arg5 Any arg5
- target [Object](#) The target (this object) to invoke the callback, can be null

## Examples

```
node.once(cc.Node.EventType.ANCHOR_CHANGED, callback);
```

off

Removes the callback previously registered with the same type, callback, target and or useCapture. This method is merely an alias to removeEventListener.

meta	description
Defined in	<a href="#">cocos2d/core/CCNode.js:1550</a>

## Parameters

- type [String](#) A string representing the event type being removed.
- callback [Function](#) The callback to remove.
- target [Object](#) The target (this object) to invoke the callback, if it's not given, only callback without target will be removed
- useCapture [Boolean](#) When set to true, the listener will be triggered at capturing phase which is ahead of the final target emit, otherwise it will be triggered during bubbling phase.

## Examples

```
this.node.off(cc.Node.EventType.TOUCH_START, this.memberFunction, this);
node.off(cc.Node.EventType.TOUCH_START, callback, this.node);
node.off(cc.Node.EventType.ANCHOR_CHANGED, callback, this);
```

targetOff

Removes all callbacks previously registered with the same target.

meta	description
Defined in	<a href="#">cocos2d/core/CCNode.js:1635</a>

## Parameters

- target [Object](#) The target to be searched for all related callbacks

## Examples

```
node.targetOff(target);
```

## hasEventListener

Checks whether the EventTarget object has any callback registered for a specific type of event.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:1679</a>

## Parameters

- type [String](#) The type of event.

## emit

Trigger an event directly with the event name and necessary arguments.

meta	description
Defined in	<a href="#">cocos2d/core/CCNode.js:1697</a>

## Parameters

- type [String](#) event type
- arg1 Any First argument in callback
- arg2 Any Second argument in callback
- arg3 Any Third argument in callback
- arg4 Any Fourth argument in callback
- arg5 Any Fifth argument in callback

## Examples

```
eventTarget.emit('fire', event);
eventTarget.emit('fire', message, emitter);
```

## dispatchEvent

Dispatches an event into the event flow. The event target is the EventTarget object upon which the dispatchEvent() method is called.

meta	description
Defined in	<a href="#">cocos2d/core/CCNode.js:1721</a>

## Parameters

- `event` [Event](#) The Event object that is dispatched into the event flow

## pauseSystemEvents

Pause node related system events registered with the current Node. Node system events includes touch and mouse events. If recursive is set to true, then this API will pause the node system events for the node and all nodes in its sub node tree.

Reference: [http://cocos2d-x.org/docs/editors\\_and\\_tools/creator-chapters/scripting/internal-events/](http://cocos2d-x.org/docs/editors_and_tools/creator-chapters/scripting/internal-events/)

meta	description
Defined in	<a href="#">cocos2d/core/CCNode.js:1735</a>

## Parameters

- `recursive` [Boolean](#) Whether to pause node system events on the sub node tree.

## Examples

```
node.pauseSystemEvents(true);
```

## resumeSystemEvents

Resume node related system events registered with the current Node. Node system events includes touch and mouse events. If recursive is set to true, then this API will resume the node system events for the node and all nodes in its sub node tree.

Reference: [http://cocos2d-x.org/docs/editors\\_and\\_tools/creator-chapters/scripting/internal-events/](http://cocos2d-x.org/docs/editors_and_tools/creator-chapters/scripting/internal-events/)

meta	description
Defined in	<a href="#">cocos2d/core/CCNode.js:1751</a>

## Parameters

- `recursive Boolean` Whether to resume node system events on the sub node tree.

## Examples

```
node.resumeSystemEvents(true);
```

### \_getCapturingTargets

Get all the targets listening to the supplied type of event in the target's capturing phase. The capturing phase comprises the journey from the root to the last node BEFORE the event target's node. The result should save in the array parameter, and MUST SORT from child nodes to parent nodes.

Subclasses can override this method to make event propagable.

meta	description
Defined in	<a href="#">cocos2d/core/CCNode.js:1813</a>

## Parameters

- `type String` the event type
- `array Array` the array to receive targets

## Examples

```
-----
Subclasses can override this method to make event propagable
```js
for (var target = this._parent; target; target = target._parent) {
    if (target._capturingListeners &&
        target._capturingListeners.hasEventListener(type)) {
        array.push(target);
    }
}
```

### \_getBubblingTargets

Get all the targets listening to the supplied type of event in the target's bubbling phase. The bubbling phase comprises any SUBSEQUENT nodes encountered on the return trip to the root of the tree. The result should save in the array parameter, and MUST SORT from child nodes to parent nodes.

Subclasses can override this method to make event propagable.

meta	description
Defined in	<a href="#">cocos2d/core/CCNode.js:1835</a>

## Parameters

- type [String](#) the event type
- array [Array](#) the array to receive targets

## runAction

Executes an action, and returns the action that is executed.

The node becomes the action's target. Refer to cc.Action's `getTarget()`  
Calling `runAction` while the node is not active won't have any effect.

Note : You shouldn't modify the action after `runAction`, that won't take any effect.  
if you want to modify, when you define action plus.

meta	description
Returns	<a href="#">Action</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:1857</a>

## Parameters

- action [Action](#)

## Examples

```
var action = cc.scaleTo(0.2, 1, 0.6);
node.runAction(action);
node.runAction(action).repeatForever(); // fail
node.runAction(action.repeatForever()); // right
```

## pauseAllActions

Pause all actions running on the current node. Equals to `cc.director.getActionManager().pauseTarget(node)`.

meta	description
Defined in	<a href="#">cocos2d/core/CCNode.js:1886</a>

## Examples

```
node.pauseAllActions();
```

### resumeAllActions

Resume all paused actions on the current node. Equals to `cc.director.getActionManager().resumeTarget(node)`.

meta	description
Defined in	<a href="#">cocos2d/core/CCNode.js:1897</a>

## Examples

```
node.resumeAllActions();
```

### stopAllActions

Stops and removes all actions from the running action list .

meta	description
Defined in	<a href="#">cocos2d/core/CCNode.js:1908</a>

## Examples

```
node.stopAllActions();
```

### stopAction

Stops and removes an action from the running action list.

meta	description
Defined in	<a href="#">cocos2d/core/CCNode.js:1919</a>

## Parameters

- `action` [Action](#) An action object to be removed.

## Examples

```
var action = cc.scaleTo(0.2, 1, 0.6);
node.stopAction(action);
```

### stopActionByTag

Removes an action from the running action list by its tag.

meta	description
Defined in	<a href="#">cocos2d/core/CCNode.js:1932</a>

## Parameters

- `tag` [Number](#) A tag that indicates the action to be removed.

## Examples

```
node.stopAction(1);
```

### getActionByTag

Returns an action from the running action list by its tag.

meta	description
Returns	<a href="#">Action</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:1948</a>

## Parameters

- `tag` [Number](#)

## Examples

```
var action = node.getActionByTag(1);
```

### getNumberOfRunningActions

Returns the numbers of actions that are running plus the ones that are scheduled to run (actions in actionsToAdd and actions arrays).

Composable actions are counted as 1 action. Example:

If you are running 1 Sequence of 7 actions, it will return 1.  
If you are running 7 Sequences of 2 actions, it will return 7.</p>

meta	description
Returns	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:1968</a>

## Examples

```
var count = node.getNumberOfRunningActions();
cc.log("Running Action Count: " + count);
```

## getPosition

Returns a copy of the position (x, y) of the node in its parent's coordinates.

meta	description
Returns	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:1994</a>

## Examples

```
cc.log("Node Position: " + node.getPosition());
```

## setPosition

Sets the position (x, y) of the node in its parent's coordinates.  
Usually we use cc.v2(x, y) to compose cc.Vec2 object.  
and Passing two numbers (x, y) is more efficient than passing cc.Vec2 object.

meta	description
Defined in	<a href="#">cocos2d/core/CCNode.js:2006</a>

## Parameters

- `newPosOrX` [Vec2](#) | [Number](#) X coordinate for position or the position (x, y) of the node in coordinates
- `y` [Number](#) Y coordinate for position

## Examples

```
node.setPosition(cc.v2(0, 0));
node.setPosition(0, 0);
```

## getScale

Returns the scale factor of the node. Assertion will fail when scale x != scale y.

meta	description
Returns	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:2066</a>

## Examples

```
cc.log("Node Scale: " + node.getScale());
```

## setScale

Sets the scale factor of the node. 1.0 is the default scale factor. This function can modify the X and Y scale at the same time.

meta	description
Defined in	<a href="#">cocos2d/core/CCNode.js:2082</a>

## Parameters

- `scaleX` [Number](#) | [Vec2](#) scaleX or scale
- `scaleY` [Number](#)

## Examples

```
node.setScale(cc.v2(1, 1));
node.setScale(1);
```

## setRotation

Set rotation of node (along z axi).

meta	description
Defined in	<a href="#">cocos2d/core/CCNode.js:2112</a>

### Parameters

- `rotation Number` Degree rotation value

## getRotation

Get rotation of node (along z axi).

meta	description
Defined in	<a href="#">cocos2d/core/CCNode.js:2119</a>

### Parameters

- `rotation Number` Degree rotation value

## getContentSize

Returns a copy the untransformed size of the node.

The contentSize remains the same no matter the node is scaled or rotated.

All nodes has a size. Layer and Scene has the same size of the screen by default.

meta	description
Returns	<a href="#">Size</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:2126</a>

### Examples

```
cc.log("Content Size: " + node.getContentSize());
```

## setContentSize

Sets the untransformed size of the node.

The contentSize remains the same no matter the node is scaled or rotated.  
All nodes has a size. Layer and Scene has the same size of the screen.

meta	description
Defined in	<a href="#">cocos2d/core/CCNode.js:2141</a>

### Parameters

- `size Size | Number` The untransformed size of the node or The untransformed size's width of the node.
- `height Number` The untransformed size's height of the node.

### Examples

```
node.setContentSize(cc.size(100, 100));
node.setContentSize(100, 100);
```

## getAnchorPoint

Returns a copy of the anchor point.

Anchor point is the point around which all transformations and positioning manipulations take place.

It's like a pin in the node where it is "attached" to its parent.

The anchorPoint is normalized, like a percentage. (0,0) means the bottom-left corner and (1,1) means the top-right corner.

But you can use values higher than (1,1) and lower than (0,0) too.

The default anchor point is (0.5,0.5), so it starts at the center of the node.

meta	description
Returns	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:2184</a>

### Examples

```
cc.log("Node AnchorPoint: " + node.getAnchorPoint());
```

## setAnchorPoint

Sets the anchor point in percent.

anchor point is the point around which all transformations and positioning manipulations take place.

It's like a pin in the node where it is "attached" to its parent.

The anchorPoint is normalized, like a percentage. (0,0) means the bottom-left corner and (1,1) means the top-right corner.

But you can use values higher than (1,1) and lower than (0,0) too.

The default anchor point is (0.5,0.5), so it starts at the center of the node.

meta	description
Defined in	<a href="#">cocos2d/core/CCNode.js:2208</a>

### Parameters

- `point Vec2 | Number` The anchor point of node or The x axis anchor of node.
- `y Number` The y axis anchor of node.

### Examples

```
node.setAnchorPoint(cc.v2(1, 1));
node.setAnchorPoint(1, 1);
```

## lookAt

Set rotation by lookAt target point, normally used by Camera Node

meta	description
Defined in	<a href="#">cocos2d/core/CCNode.js:2383</a>

### Parameters

- `pos Vec3`
- `up Vec3` default is (0,1,0)

## getLocalMatrix

Get the local transform matrix (4x4), based on parent node coordinates

meta	description
Returns	vmath.Mat4
Defined in	<a href="#">cocos2d/core/CCNode.js:2526</a>

## Parameters

- out vmath.Mat4 The matrix object to be filled with data

## Examples

```
let mat4 = vmath.mat4.create();
node.getLocalMatrix(mat4);
```

## getWorldMatrix

Get the world transform matrix (4x4)

meta	description
Returns	vmath.Mat4
Defined in	<a href="#">cocos2d/core/CCNode.js:2542</a>

## Parameters

- out vmath.Mat4 The matrix object to be filled with data

## Examples

```
let mat4 = vmath.mat4.create();
node.getWorldMatrix(mat4);
```

## convertToNodeSpace

Converts a Point to node (local) space coordinates then add the anchor point position. So the return position will be related to the left bottom corner of the node's bounding box. This equals to the API behavior of cocos2d-x, you probably want to use convertToNodeSpaceAR instead

meta	description
Returns	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:2558</a>

## Parameters

- worldPoint [Vec2](#)

## Examples

```
var newVec2 = node.convertToWorldSpace(cc.v2(100, 100));
```

## convertToWorldSpace

Converts a Point related to the left bottom corner of the node's bounding box to world space coordinates. This equals to the API behavior of cocos2d-x, you probably want to use convertToWorldSpaceAR instead

meta	description
Returns	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:2581</a>

## Parameters

- nodePoint [Vec2](#)

## Examples

```
var newVec2 = node.convertToWorldSpace(cc.v2(100, 100));
```

## convertToNodeSpaceAR

Converts a Point to node (local) space coordinates in which the anchor point is the origin position.

meta	description
Returns	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:2601</a>

## Parameters

- worldPoint [Vec2](#)

## Examples

```
var newVec2 = node.convertToWorldSpaceAR(cc.v2(100, 100));
```

## convertToWorldSpaceAR

Converts a Point in node coordinates to world space coordinates.

meta	description
Returns	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:2619</a>

## Parameters

- nodePoint [Vec2](#)

## Examples

```
var newVec2 = node.convertToWorldSpaceAR(cc.v2(100, 100));
```

## getNodeToParentTransform

Returns the matrix that transform the node's (local) space coordinates into the parent's space coordinates.

The matrix is in Pixels.

meta	description
Returns	<a href="#">AffineTransform</a>

meta	description
Defined in	<a href="#">cocos2d/core/CCNode.js:2637</a>
Deprecated	since v2.0

## Parameters

- `out` [AffineTransform](#) The affine transform object to be filled with data

## Examples

```
let affineTransform = cc.AffineTransform.create();
node.getNodeToParentTransform(affineTransform);
```

## getNodeToParentTransformAR

Returns the matrix that transform the node's (local) space coordinates into the parent's space coordinates.

The matrix is in Pixels.

This method is AR (Anchor Relative).

meta	description
Returns	<a href="#">AffineTransform</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:2665</a>
Deprecated	since v2.0

## Parameters

- `out` [AffineTransform](#) The affine transform object to be filled with data

## Examples

```
let affineTransform = cc.AffineTransform.create();
node.getNodeToParentTransformAR(affineTransform);
```

## getNodeToWorldTransform

Returns the world affine transform matrix. The matrix is in Pixels.

meta	description
Returns	<a href="#">AffineTransform</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:2690</a>
Deprecated	since v2.0

## Parameters

- `out` [AffineTransform](#) The affine transform object to be filled with data

## Examples

```
let affineTransform = cc.AffineTransform.create();
node.getNodeToWorldTransform(affineTransform);
```

## getNodeToWorldTransformAR

Returns the world affine transform matrix. The matrix is in Pixels.  
This method is AR (Anchor Relative).

meta	description
Returns	<a href="#">AffineTransform</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:2717</a>
Deprecated	since v2.0

## Parameters

- `out` [AffineTransform](#) The affine transform object to be filled with data

## Examples

```
let affineTransform = cc.AffineTransform.create();
node.getNodeToWorldTransformAR(affineTransform);
```

## getParentToNodeTransform

Returns the matrix that transform parent's space coordinates to the node's (local) space coordinates.

The matrix is in Pixels. The returned transform is readonly and cannot be changed.

meta	description
Returns	<a href="#">AffineTransform</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:2740</a>
Deprecated	since v2.0

### Parameters

- `out` [AffineTransform](#) The affine transform object to be filled with data

### Examples

```
let affineTransform = cc.AffineTransform.create();
node.getParentToNodeTransform(affineTransform);
```

## getWorldToNodeTransform

Returns the inverse world affine transform matrix. The matrix is in Pixels.

返回世界坐标系到节点坐标系的逆矩阵。

meta	description
Returns	<a href="#">AffineTransform</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:2764</a>
Deprecated	since v2.0

### Parameters

- `out` [AffineTransform](#) The affine transform object to be filled with data

## Examples

```
let affineTransform = cc.AffineTransform.create();
node.getWorldToNodeTransform(affineTransform);
```

### convertTouchToNodeSpace

convenience methods which take a cc.Touch instead of cc.Vec2.

meta	description
Returns	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:2784</a>
Deprecated	since v2.0

### Parameters

- touch [Touch](#) The touch object

## Examples

```
var newVec2 = node.convertTouchToNodeSpace(touch);
```

### convertTouchToNodeSpaceAR

converts a cc.Touch (world coordinates) into a local coordinate. This method is AR (Anchor Relative).

meta	description
Returns	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:2798</a>
Deprecated	since v2.0

### Parameters

- touch [Touch](#) The touch object

## Examples

```
var newVec2 = node.convertTouchToNodeSpaceAR(touch);
```

### getBoundingBox

Returns a "local" axis aligned bounding box of the node.  
The returned box is relative only to its parent.

meta	description
Returns	<a href="#">Rect</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:2812</a>

## Examples

```
var boundingBox = node.getBoundingBox();
```

### getBoundingBoxToWorld

Returns a "world" axis aligned bounding box of the node.  
The bounding box contains self and active children's world bounding box.

meta	description
Returns	<a href="#">Rect</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:2834</a>

## Examples

```
var newRect = node.getBoundingBoxToWorld();
```

### addChild

Adds a child to the node with z order and name.

meta	description
Defined in	<a href="#">cocos2d/core/CCNode.js:2890</a>

## Parameters

- `child` [Node](#) A child node
- `zIndex` [Number](#) Z order for drawing priority. Please refer to `zIndex` property
- `name` [String](#) A name to identify the node easily. Please refer to `name` property

## Examples

```
node.addChild(newNode, 1, "node");
```

### cleanup

Stops all running actions and schedulers.

meta	description
Defined in	<a href="#">cocos2d/core/CCNode.js:2920</a>

## Examples

```
node.cleanup();
```

### sortAllChildren

Sorts the children array depends on children's `zIndex` and `arrivalOrder`, normally you won't need to invoke this function.

meta	description
Defined in	<a href="#">cocos2d/core/CCNode.js:2942</a>

### getDisplayedOpacity

Returns the displayed opacity of Node, the difference between displayed opacity and opacity is that displayed opacity is calculated based on opacity and parent node's opacity when cascade opacity enabled.

meta	description
Returns	<a href="#">number</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:3059</a>

meta	description
Deprecated	since v2.0, please use opacity property, cascade opacity is removed

### getDisplayedColor

Returns the displayed color of Node, the difference between displayed color and color is that displayed color is calculated based on color and parent node's color when cascade color enabled.

meta	description
Returns	<a href="#">Color</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:3073</a>
Deprecated	since v2.0, please use color property, cascade color is removed

### isCascadeOpacityEnabled

Cascade opacity is removed from v2.0 Returns whether node's opacity value affect its child nodes.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:3097</a>
Deprecated	since v2.0

### setCascadeOpacityEnabled

Cascade opacity is removed from v2.0 Enable or disable cascade opacity, if cascade enabled, child nodes' opacity will be the multiplication of parent opacity and its own opacity.

meta	description
Defined in	<a href="#">cocos2d/core/CCNode.js:3107</a>
Deprecated	since v2.0

## Parameters

- `cascadeOpacityEnabled Boolean`

`setOpacityModifyRGB`

Opacity modify RGB have been removed since v2.0 Set whether color should be changed with the opacity value, useless in ccsg.Node, but this function is override in some class to have such behavior.

meta	description
Defined in	<a href="#">cocos2d/core/CCNode.js:3117</a>
Deprecated	since v2.0

## Parameters

- `opacityValue Boolean`

`isOpacityModifyRGB`

Opacity modify RGB have been removed since v2.0 Get whether color should be changed with the opacity value.

meta	description
Returns	<code>Boolean</code>
Defined in	<a href="#">cocos2d/core/CCNode.js:3128</a>
Deprecated	since v2.0

## attr

Properties configuration function

All properties in attrs will be set to the node, when the setter of the node is available, the property will be set via setter function.

meta	description
Defined in	<a href="#">cocos2d/core/utils/base-node.js:405</a>

### Parameters

- attrs [Object](#) Properties to be set to node

### Examples

```
var attrs = { key: 0, num: 100 };
node.attr(attrs);
```

## getChildByUuid

Returns a child from the container given its uuid.

meta	description
Returns	<a href="#">Node</a>
Defined in	<a href="#">cocos2d/core/utils/base-node.js:424</a>

### Parameters

- uuid [String](#) The uuid to find the child node.

### Examples

```
var child = node.getChildByUuid(uuid);
```

## getChildByName

Returns a child from the container given its name.

meta	description
Returns	<a href="#">Node</a>
Defined in	<a href="#">cocos2d/core/utils/base-node.js:447</a>

## Parameters

- name [String](#) A name to find the child node.

## Examples

```
var child = node.getChildByName("Test Node");
```

## insertChild

Inserts a child to the node at a specified index.

meta	description
Defined in	<a href="#">cocos2d/core/utils/base-node.js:485</a>

## Parameters

- child [Node](#) the child node to be inserted
- siblingIndex [Number](#) the sibling index to place the child in

## Examples

```
node.insertChild(child, 2);
```

## getSiblingIndex

Get the sibling index.

meta	description
Returns	<a href="#">number</a>
Defined in	<a href="#">cocos2d/core/utils/base-node.js:503</a>

## Examples

```
var index = node.getSiblingIndex();
```

### setSiblingIndex

Set the sibling index of this node.

meta	description
Defined in	<a href="#">cocos2d/core/utils/base-node.js:520</a>

## Parameters

- `index` [Number](#)

## Examples

```
node.setSiblingIndex(1);
```

### walk

Walk though the sub children tree of the current node. Each node, including the current node, in the sub tree will be visited two times, before all children and after all children. This function call is not recursive, it's based on stack. Please don't walk any other node inside the walk process.

meta	description
Defined in	<a href="#">cocos2d/core/utils/base-node.js:551</a>

## Parameters

- `prefunc` [Function](#) The callback to process node when reach the node for the first time
- `target` [BaseNode](#) The current visiting node
- `postfunc` [Function](#) The callback to process node when re-visit the node after walked all children in its sub tree
- `target` [BaseNode](#) The current visiting node

## Examples

```
node.walk(function (target) {  
    console.log('Walked through node ' + target.name + ' for the first time');  
}, function (target) {
```

```
    console.log('Walked through node ' + target.name + ' after walked all children  
in its sub tree');  
});
```

## removeFromParent

Remove itself from its parent node. If cleanup is `true`, then also remove all events and actions.

If the cleanup parameter is not passed, it will force a cleanup, so it is recommended that you always pass in the `false` parameter when calling this API.

If the node orphan, then nothing happens.

meta	description
Defined in	<a href="#">cocos2d/core/utils/base-node.js:665</a>

### Parameters

- `cleanup Boolean` true if all actions and callbacks on this node should be removed, false otherwise.

## Examples

```
node.removeFromParent();  
node.removeFromParent(false);
```

## removeChild

Removes a child from the container. It will also cleanup all running actions depending on the cleanup parameter. If the cleanup parameter is not passed, it will force a cleanup.

"remove" logic MUST only be on this method

If a class wants to extend the 'removeChild' behavior it only needs to override this method.

meta	description
Defined in	<a href="#">cocos2d/core/utils/base-node.js:689</a>

### Parameters

- `child Node` The child node which will be removed.
- `cleanup Boolean` true if all running actions and callbacks on the child node will be cleanup, false otherwise.

## Examples

```
node.removeChild(newNode);
node.removeChild(newNode, false);
```

### removeAllChildren

Removes all children from the container and do a cleanup all running actions depending on the cleanup parameter.

If the cleanup parameter is not passed, it will force a cleanup.

meta	description
Defined in	<a href="#">cocos2d/core/utils/base-node.js:717</a>

## Parameters

- `cleanup Boolean` true if all running actions on all children nodes should be cleanup, false otherwise.

## Examples

```
node.removeAllChildren();
node.removeAllChildren(false);
```

### isChildOf

Is this node a child of the given node?

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/utils/base-node.js:748</a>

## Parameters

- `parent Node`

## Examples

```
node.isChildOf(newNode);
```

## getComponent

Returns the component of supplied type if the node has one attached, null if it doesn't.

You can also get component in the node by passing in the name of the script.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/utils/base-node.js:771</a>

### Parameters

- `typeOrClassName` [Function](#) | [String](#)

### Examples

```
// get sprite component.  
var sprite = node.getComponent(cc.Sprite);  
// get custom test calss.  
var test = node.getComponent("Test");
```

## getComponents

Returns all components of supplied type in the node.

meta	description
Returns	<a href="#">Component[]</a>
Defined in	<a href="#">cocos2d/core/utils/base-node.js:798</a>

### Parameters

- `typeOrClassName` [Function](#) | [String](#)

### Examples

```
var sprites = node.getComponents(cc.Sprite);  
var tests = node.getComponents("Test");
```

## getComponentInChildren

Returns the component of supplied type in any of its children using depth first search.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/utils/base-node.js:819</a>

### Parameters

- `typeOrClassName` [Function](#) | [String](#)

### Examples

```
var sprite = node.getComponentInChildren(cc.Sprite);
var Test = node.getComponentInChildren("Test");
```

## getComponentsInChildren

Returns all components of supplied type in self or any of its children.

meta	description
Returns	<a href="#">Component[]</a>
Defined in	<a href="#">cocos2d/core/utils/base-node.js:840</a>

### Parameters

- `typeOrClassName` [Function](#) | [String](#)

### Examples

```
var sprites = node.getComponentsInChildren(cc.Sprite);
var tests = node.getComponentsInChildren("Test");
```

## addComponent

Adds a component class to the node. You can also add component to node by passing in the name of the script.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/utils/base-node.js:876</a>

## Parameters

- `typeOrClassName` [Function](#) | [String](#) The constructor or the class name of the component to add

## Examples

```
var sprite = node.addComponent(cc.Sprite);
var test = node.addComponent("Test");
```

## \_addComponentAt

This api should only used by undo system

meta	description
Defined in	<a href="#">cocos2d/core/utils/base-node.js:965</a>

## Parameters

- `comp` [Component](#)
- `index` [Number](#)

## removeComponent

Removes a component identified by the given name or removes the component object given. You can also use component.destroy() if you already have the reference.

meta	description
Defined in	<a href="#">cocos2d/core/utils/base-node.js:1013</a>
Deprecated	please destroy the component to remove it.

## Parameters

- **component** [String](#) | [Function](#) | [Component](#) The need remove component.

## Examples

```
node.removeComponent(cc.Sprite);
var Test = require("Test");
node.removeComponent(Test);
```

## \_getDependComponent

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/utils/base-node.js:1041</a>

## Parameters

- **depended** [Component](#)

## destroyAllChildren

Destroy all children from the node, and release all their own references to other objects.

Actual destruct operation will delayed until before rendering.

meta	description
Defined in	<a href="#">cocos2d/core/utils/base-node.js:1108</a>

## Examples

```
node.destroyAllChildren();
```

## destroy

Destroy this Object, and release all its own references to other objects.

Actual object destruction will delayed until before rendering. From the next frame, this object is not usable any more. You can use cc.isValid(obj) to check whether the object is destroyed before accessing it.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:296</a>

## Examples

```
obj.destroy();
```

### \_destruct

Clear all references in the instance.

NOTE: this method will not clear the getter or setter functions which defined in the instance of CCObject. You can override the \_destruct method if you need, for example:

```
_destruct: function () { for (var key in this) { if (this.hasOwnProperty(key)) { switch (typeof this[key]) { case 'string': this[key] = ""; break; case 'object': case 'function': this[key] = null; break; } } }
```

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:430</a>

### \_onPreDestroy

Called before the object being destroyed.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:463</a>

### \_serialize

The customized serialization for this object. (Editor Only)

meta	description
Returns	<a href="#">object</a>

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:488</a>

## Parameters

- `exporting Boolean`

`_deserialize`

Init this object from the custom serialized data.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:498</a>

## Parameters

- `data Object` the serialized json data
- `ctx _Deserializer`

## *Events*

### **position-changed Event**

Module: [cc](#)

## **Index**

## **Details**

### **size-changed Event**

Module: [cc](#)

## **Index**

## **Details**

### **anchor-changed Event**

Module: [cc](#)

## Index

### Details

#### **child-added Event**

Module: [cc](#)

## Index

### Details

#### **child-removed Event**

Module: [cc](#)

## Index

### Details

#### **child-reorder Event**

Module: [cc](#)

## Index

### Details

#### **group-changed Event**

Module: [cc](#)

## Index

### Details

#### **active-in-hierarchy-changed Event**

Module: [cc](#)

Note: This event is only emitted from the top most node whose active value did changed, not including its child nodes.

## **Node.EventType Class**

Module: [cc](#)

The event type supported by Node

## Index

### Properties

- `TOUCH_START` String The event type for touch start event, you can use its value directly: 'touchstart'
- `TOUCH_MOVE` String The event type for touch move event, you can use its value directly: 'touchmove'
- `TOUCH_END` String The event type for touch end event, you can use its value directly: 'touchend'
- `TOUCH_CANCEL` String The event type for touch cancel event, you can use its value directly: 'touchcancel'
- `MOUSE_DOWN` String The event type for mouse down events, you can use its value directly: 'mousedown'
- `MOUSE_MOVE` String The event type for mouse move events, you can use its value directly: 'mousemove'
- `MOUSE_ENTER` String The event type for mouse enter target events, you can use its value directly: 'mouseenter'
- `MOUSE_LEAVE` String The event type for mouse leave target events, you can use its value directly: 'mouseleave'
- `MOUSE_UP` String The event type for mouse up events, you can use its value directly: 'mouseup'
- `MOUSE_WHEEL` String The event type for mouse wheel events, you can use its value directly: 'mousewheel'
- `POSITION_CHANGED` String The event type for position change events.
- `ROTATION_CHANGED` String The event type for rotation change events.
- `SCALE_CHANGED` String The event type for scale change events.
- `SIZE_CHANGED` String The event type for size change events.
- `ANCHOR_CHANGED` String The event type for anchor point change events.
- `CHILD_ADDED` String The event type for new child added events.
- `CHILD_REMOVED` String The event type for child removed events.
- `CHILD_REORDER` String The event type for children reorder events.
- `GROUP_CHANGED` String The event type for node group changed events.

## Details

### Properties

#### TOUCH\_START

The event type for touch start event, you can use its value directly: 'touchstart'

meta	description
Type	<a href="#">String</a>

<b>meta</b>	<b>description</b>
Defined in	<a href="#">cocos2d/core/CCNode.js:134</a>

## TOUCH\_MOVE

The event type for touch move event, you can use its value directly: 'touchmove'

<b>meta</b>	<b>description</b>
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:141</a>

## TOUCH\_END

The event type for touch end event, you can use its value directly: 'touchend'

<b>meta</b>	<b>description</b>
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:148</a>

## TOUCH\_CANCEL

The event type for touch end event, you can use its value directly: 'touchcancel'

<b>meta</b>	<b>description</b>
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:155</a>

## MOUSE\_DOWN

The event type for mouse down events, you can use its value directly: 'mousedown'

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:163</a>

## MOUSE\_MOVE

The event type for mouse move events, you can use its value directly: 'mousemove'

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:170</a>

## MOUSE\_ENTER

The event type for mouse enter target events, you can use its value directly: 'mouseenter'

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:177</a>

## MOUSE\_LEAVE

The event type for mouse leave target events, you can use its value directly: 'mouseleave'

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:184</a>

## MOUSE\_UP

The event type for mouse up events, you can use its value directly: 'mouseup'

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:191</a>

## MOUSE\_WHEEL

The event type for mouse wheel events, you can use its value directly: 'mousewheel'

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:198</a>

## POSITION\_CHANGED

The event type for position change events. Performance note, this event will be triggered every time corresponding properties being changed, if the event callback have heavy logic it may have great performance impact, try to avoid such scenario.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:206</a>

## ROTATION\_CHANGED

The event type for rotation change events. Performance note, this event will be triggered every time corresponding properties being changed, if the event callback have heavy logic it may have great performance impact, try to avoid such scenario.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:216</a>

### SCALE\_CHANGED

The event type for scale change events. Performance note, this event will be triggered every time corresponding properties being changed, if the event callback have heavy logic it may have great performance impact, try to avoid such scenario.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:226</a>

### SIZE\_CHANGED

The event type for size change events. Performance note, this event will be triggered every time corresponding properties being changed, if the event callback have heavy logic it may have great performance impact, try to avoid such scenario.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:236</a>

### ANCHOR\_CHANGED

The event type for anchor point change events. Performance note, this event will be triggered every time corresponding properties being changed, if the event callback have heavy logic it may have great performance impact, try to avoid such scenario.

meta	description
Type	<a href="#">String</a>

<b>meta</b>	<b>description</b>
Defined in	<a href="#">cocos2d/core/CCNode.js:246</a>

## CHILD\_ADDED

The event type for new child added events.

<b>meta</b>	<b>description</b>
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:256</a>

## CHILD\_REMOVED

The event type for child removed events.

<b>meta</b>	<b>description</b>
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:263</a>

## CHILD\_reordered

The event type for children reorder events.

<b>meta</b>	<b>description</b>
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:270</a>

## GROUP\_CHANGED

The event type for node group changed events.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/CCNode.js:277</a>

## NodePool Class

Module: [cc](#)

cc.NodePool is the cache pool designed for node type.

It can help you to improve your game performance for objects which need frequent release and recreate operations

It's recommended to create cc.NodePool instances by node type, the type corresponds to node type in game design, not the class, for example, a prefab is a specific node type.

When you create a node pool, you can pass a Component which contains unuse, reuse functions to control the content of node.

Some common use case is :

- 1. Bullets in game (die very soon, massive creation and recreation, no side effect on other objects)<br/>
- 2. Blocks in candy crash (massive creation and recreation)<br/>etc...

## Index

### Properties

- `poolHandlerComp` Function|String The pool handler component, it could be the class name or the constructor.

### Methods

- `constructor` Constructor for creating a pool for a specific node template (usually a prefab).
- `size` The current available size in the pool
- `clear` Destroy all cached nodes in the pool
- `put` Put a new Node into the pool.
- `get` Get a obj from pool, if no available object in pool, null will be returned.

## Details

## Properties

### poolHandlerComp

The pool handler component, it could be the class name or the constructor.

meta	description
Type	<a href="#">Function   String</a>
Defined in	<a href="#">extensions/ccpool/CCNodePool.js:76</a>

## Methods

### constructor

Constructor for creating a pool for a specific node template (usually a prefab). You can pass a component (type or name) argument for handling event for reusing and recycling node.

meta	description
Defined in	<a href="#">extensions/ccpool/CCNodePool.js:57</a>

### Parameters

- `poolHandlerComp` [Function | String](#) !#en The constructor or the class name of the component to control the unuse/reuse logic. !#zh 处理节点回收和复用事件逻辑的组件类型或名称。

### Examples

```
properties: {
    template: cc.Prefab
},
onLoad () {
// MyTemplateHandler is a component with 'unuse' and 'reuse' to handle events when node is reused
or recycled.
    this.myPool = new cc.NodePool('MyTemplateHandler');
}
```

### size

The current available size in the pool

meta	description
Returns	<a href="#">Number</a>
Defined in	<a href="#">extensions/ccpool/CCNodePool.js:88</a>

## clear

Destroy all cached nodes in the pool

meta	description
Defined in	<a href="#">extensions/ccpool/CCNodePool.js:98</a>

## put

Put a new Node into the pool. It will automatically remove the node from its parent without cleanup. It will also invoke unuse method of the poolHandlerComp if exist.

meta	description
Defined in	<a href="#">extensions/ccpool/CCNodePool.js:111</a>

## Parameters

- obj [Node](#)

## Examples

```
let myNode = cc.instantiate(this.template);
this.myPool.put(myNode);
```

## get

Get a obj from pool, if no available object in pool, null will be returned. This function will invoke the reuse function of poolHandlerComp if exist.

meta	description
Returns	<a href="#">Node</a>   Null

meta	description
Defined in	<a href="#">extensions/ccpool/CCNodePool.js:139</a>

## Parameters

- `params` Any !#en Params to pass to 'reuse' method in poolHandlerComp !#zh 向 poolHandlerComp 中的 'reuse' 函数传递的参数

## Examples

```
let newNode = this.myPool.get();
```

## OriginalContainer Class

Extends [ContainerStrategy](#)

Module: [decorator](#) Parent Module: [cc](#)

## Index

### Methods

- `preApply` Manipulation before applying the strategy
- `apply` Function to apply this strategy
- `postApply` Manipulation after applying the strategy

## Details

### Methods

#### preApply

Manipulation before applying the strategy

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCView.js:1031</a>

## Parameters

- `view` [View](#) The target view

apply

Function to apply this strategy

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCView.js:1041</a>

## Parameters

- `view` [View](#)
- `designedResolution` [Size](#)

postApply

Manipulation after applying the strategy

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCView.js:1052</a>

## Parameters

- `view` [View](#) The target view

# PageView Class

Extends [ScrollView](#)

Module: [cc](#)

The PageView control

## Index

### Properties

- `sizeMode` `PageView.SizeMode` Specify the size type of each page in PageView.

- `direction` `PageView.Direction` The page view direction
- `scrollThreshold` `Number` The scroll threshold value, when drag exceeds this value,...
- `autoPageTurningThreshold` `Number` Auto page turning velocity threshold.
- `pageTurningEventTiming` `Number` Change the PageTurning event timing of PageView.
- `indicator` `PageViewIndicator` The Page View Indicator
- `pageTurningSpeed` `Number` The time required to turn over a page.
- `pageEvents` `Component.EventHandler[]` PageView events callback
- `content` `Node` This is a reference to the UI element to be scrolled.
- `horizontal` `Boolean` Enable horizontal scroll.
- `vertical` `Boolean` Enable vertical scroll.
- `inertia` `Boolean` When inertia is set, the content will continue to move when touch ended.
- `brake` `Number` It determines how quickly the content stop moving.
- `elastic` `Boolean` When elastic is set, the content will be bounce back when move out of boundary.
- `bounceDuration` `Number` The elapse time of bouncing back.
- `horizontalScrollBar` `Scrollbar` The horizontal scrollbar reference.
- `verticalScrollBar` `Scrollbar` The vertical scrollbar reference.
- `scrollEvents` `Component.EventHandler[]` Scrollview events callback
- `cancelInnerEvents` `Boolean` If cancellInnerEvents is set to true, the scroll behavior will cancel touch events on inner content nodes
- `__eventTargets` `Array` Register all related EventTargets,...
- `node` `Node` The node this component is attached to.
- `uuid` `String` The uuid for editor.
- `_enabled` `Boolean`
- `enabled` `Boolean` indicates whether this component is enabled or not.
- `enabledInHierarchy` `Boolean` indicates whether this component is enabled and its node is also active in the hierarchy.
- `_isOnLoadCalled` `Number` Returns a value which used to indicate the onLoad get called or not.
- `_name` `String`
- `_objFlags` `Number`
- `name` `String` The name of the object.
- `isValid` `Boolean` Indicates whether the object is not yet destroyed.

## Methods

- `getCurrentPageIndex` Returns current page index
- `setCurrentPageIndex` Set current page index
- `getPages` Returns all pages of pageview
- `addPage` At the end of the current page view to insert a new view
- `insertPage` Inserts a page in the specified location
- `removePage` Removes a page from PageView.
- `removePageAtIndex` Removes a page at index of PageView.
- `removeAllPages` Removes all pages from PageView
- `scrollToPage` Scroll PageView to index.
- `scrollToBottom` Scroll the content to the bottom boundary of ScrollView.
- `scrollToTop` Scroll the content to the top boundary of ScrollView.
- `scrollToLeft` Scroll the content to the left boundary of ScrollView.
- `scrollToRight` Scroll the content to the right boundary of ScrollView.
- `scrollToTopLeft` Scroll the content to the top left boundary of ScrollView.
- `scrollToTopRight` Scroll the content to the top right boundary of ScrollView.

- `scrollToBottomLeft` Scroll the content to the bottom left boundary of ScrollView.
- `scrollToBottomRight` Scroll the content to the bottom right boundary of ScrollView.
- `scrollToOffset` Scroll with an offset related to the ScrollView's top left origin, if timeInSecond is omitted, then it will jump to the
- `getScrollOffset` Get the positive offset value corresponds to the content's top left boundary.
- `getMaxScrollOffset` Get the maximize available scroll offset
- `scrollToPercentHorizontal` Scroll the content to the horizontal percent position of ScrollView.
- `scrollTo` Scroll the content to the percent position of ScrollView in any direction.
- `scrollToPercentVertical` Scroll the content to the vertical percent position of ScrollView.
- `stopAutoScroll` Stop auto scroll immediately
- `setContentPosition` Modify the content position.
- `getContentPosition` Query the content's position in its parent space.
- `isScrolling` Query whether the user is currently dragging the ScrollView to scroll it
- `isAutoScrolling` Query whether the ScrollView is currently scrolling because of a bounceback or inertia slowdown.
- `update` This is a lifecycle method.
- `lateUpdate` This is a lifecycle method.
- `__preload` \_\_preload is called before every onLoad.
- `onLoad` When attaching to an active node or its node first activated.
- `start` Called before all scripts' update if the Component is enabled the first time.
- `onEnable` This is a lifecycle method.
- `onDisable` This is a lifecycle method.
- `onDestroy` This is a lifecycle method.
- `onFocusInEditor`
- `onLostFocusInEditor`
- `resetInEditor` Called to initialize the component or node's properties when adding the component the first time or when the Reset command is used.
- `addComponent` Adds a component class to the node.
- `getComponent` Returns the component of supplied type if the node has one attached, null if it doesn't....
- `getComponents` Returns all components of supplied Type in the node.
- `getComponentInChildren` Returns the component of supplied type in any of its children using depth first search.
- `getComponentsInChildren` Returns the components of supplied type in self or any of its children using depth first search.
- `_getLocalBounds` If the component's bounding box is different from the node's, you can implement this method to supply
- `onRestore` for undo/redo operation.
- `schedule` Schedules a custom selector....
- `scheduleOnce` Schedules a callback function that runs only once, with a delay of 0 or larger.
- `unschedule` Unschedules a custom callback function.
- `unscheduleAllCallbacks` unschedule all scheduled callback functions: custom callback functions, and the 'update' callback function....
- `destroy` Actual object destruction will delayed until before rendering.
- `_destruct` Clear all references in the instance.
- `_onPreDestroy` Called before the object being destroyed.
- `_serialize` The customized serialization for this object.
- `_deserialize` Init this object from the custom serialized data.

## Events

- [page-turning](#) Note: This event is emitted from the node to which the component belongs.
- [scroll-to-top](#) Note: This event is emitted from the node to which the component belongs.
- [scroll-to-bottom](#) Note: This event is emitted from the node to which the component belongs.
- [scroll-to-left](#) Note: This event is emitted from the node to which the component belongs.
- [scroll-to-right](#) Note: This event is emitted from the node to which the component belongs.
- [scrolling](#) Note: This event is emitted from the node to which the component belongs.
- [bounce-bottom](#) Note: This event is emitted from the node to which the component belongs.
- [bounce-top](#) Note: This event is emitted from the node to which the component belongs.
- [bounce-left](#) Note: This event is emitted from the node to which the component belongs.
- [bounce-right](#) Note: This event is emitted from the node to which the component belongs.
- [scroll-ended](#) Note: This event is emitted from the node to which the component belongs.
- [touch-up](#) Note: This event is emitted from the node to which the component belongs.
- [scroll-began](#) Note: This event is emitted from the node to which the component belongs.

## Details

### Properties

#### sizeMode

Specify the size type of each page in PageView.

meta	description
Type	<a href="#">PageView.SizeMode</a>
Defined in	<a href="#">cocos2d/core/components/CCPageView.js:112</a>

#### direction

The page view direction

meta	description
Type	<a href="#">PageView.Direction</a>
Defined in	<a href="#">cocos2d/core/components/CCPageView.js:126</a>

## scrollThreshold

The scroll threshold value, when drag exceeds this value, release the next page will automatically scroll, less than the restore

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/CCPageView.js:140</a>

## autoPageTurningThreshold

Auto page turning velocity threshold. When users swipe the PageView quickly, it will calculate a velocity based on the scroll distance and time, if the calculated velocity is larger than the threshold, then it will trigger page turning.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/CCPageView.js:155</a>

## pageTurningEventTiming

Change the PageTurning event timing of PageView.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/CCPageView.js:172</a>

## indicator

The Page View Indicator

meta	description
Type	<a href="#">PageViewIndicator</a>
Defined in	<a href="#">cocos2d/core/components/CCPageView.js:184</a>

### pageTurningSpeed

The time required to turn over a page. unit: second

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/CCPageView.js:200</a>

### pageEvents

PageView events callback

meta	description
Type	<a href="#">Component.EventHandler[]</a>
Defined in	<a href="#">cocos2d/core/components/CCPageView.js:211</a>

### content

This is a reference to the UI element to be scrolled.

meta	description
Type	<a href="#">Node</a>
Defined in	<a href="#">cocos2d/core/components/CCScrollView.js:202</a>

## horizontal

Enable horizontal scroll.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCScrollView.js:213</a>

## vertical

Enable vertical scroll.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCScrollView.js:224</a>

## inertia

When inertia is set, the content will continue to move when touch ended.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCScrollView.js:235</a>

## brake

It determines how quickly the content stop moving. A value of 1 will stop the movement immediately. A value of 0 will never stop the movement until it reaches to the boundary of scrollview.

meta	description
Type	<a href="#">Number</a>

<b>meta</b>	<b>description</b>
Defined in	<a href="#">cocos2d/core/components/CCScrollView.js:245</a>

### elastic

When elastic is set, the content will be bounce back when move out of boundary.

<b>meta</b>	<b>description</b>
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCScrollView.js:260</a>

### bounceDuration

The elapse time of bouncing back. A value of 0 will bounce back immediately.

<b>meta</b>	<b>description</b>
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/CCScrollView.js:271</a>

### horizontalScrollBar

The horizontal scrollbar reference.

<b>meta</b>	<b>description</b>
Type	<a href="#">Scrollbar</a>
Defined in	<a href="#">cocos2d/core/components/CCScrollView.js:282</a>

### verticalScrollBar

The vertical scrollbar reference.

meta	description
Type	<a href="#">Scrollbar</a>
Defined in	<a href="#">cocos2d/core/components/CCScrollView.js:300</a>

## scrollEvents

Scrollview events callback

meta	description
Type	<a href="#">Component.EventHandler[]</a>
Defined in	<a href="#">cocos2d/core/components/CCScrollView.js:318</a>

## cancellInnerEvents

If cancellInnerEvents is set to true, the scroll behavior will cancel touch events on inner content nodes It's set to true by default.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCScrollView.js:329</a>

## \_\_eventTargets

Register all related EventTargets, all event callbacks will be removed in \_onPreDestroy

meta	description
Type	<a href="#">Array</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:61</a>

## node

The node this component is attached to. A component is always attached to a node.

meta	description
Type	<a href="#">Node</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:75</a>

## Examples

```
cc.log(comp.node);
```

## uuid

The uuid for editor.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:106</a>

## Examples

```
cc.log(comp.uuid);
```

## \_enabled

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:147</a>

## enabled

indicates whether this component is enabled or not.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:154</a>

## Examples

```
comp.enabled = true;
cc.log(comp.enabled);
```

### enabledInHierarchy

indicates whether this component is enabled and its node is also active in the hierarchy.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:185</a>

## Examples

```
cc.log(comp.enabledInHierarchy);
```

### \_isOnLoadCalled

Returns a value which used to indicate the onLoad get called or not.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:201</a>

## Examples

```
cc.log(this._isOnLoadCalled > 0);
```

## \_name

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:76</a>

## \_objFlags

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:83</a>

## name

The name of the object.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:243</a>

## Examples

```
obj.name = "New Obj";
```

## isValid

Indicates whether the object is not yet destroyed. (It will not be available after being destroyed)

When an object's `destroy` is called, it is actually destroyed after the end of this frame. So `isValid` will return false from the next frame, while `isValid` in the current frame will still be true. If you want to determine whether the current frame has called `destroy`, use `cc.isValid(obj, true)`, but this is often caused by a particular logical requirements, which is not normally required.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:261</a>

## Examples

```
var node = new cc.Node();
cc.log(node.isValid);    // true
node.destroy();
cc.log(node.isValid);    // true, still valid in this frame
// after a frame...
cc.log(node.isValid);    // false, destroyed in the end of last frame
```

## Methods

### getCurrentPageIndex

Returns current page index

meta	description
Returns	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/CCPageView.js:258</a>

### setCurrentPageIndex

Set current page index

meta	description
Defined in	<a href="#">cocos2d/core/components/CCPageView.js:268</a>

## Parameters

- index [Number](#)

## getPages

Returns all pages of pageview

meta	description
Returns	<a href="#">Node[]</a>
Defined in	<a href="#">cocos2d/core/components/CCPageView.js:278</a>

## addPage

At the end of the current page view to insert a new view

meta	description
Defined in	<a href="#">cocos2d/core/components/CCPageView.js:288</a>

Parameters

- `page` [Node](#)

## insertPage

Inserts a page in the specified location

meta	description
Defined in	<a href="#">cocos2d/core/components/CCPageView.js:302</a>

Parameters

- `page` [Node](#)
- `index` [Number](#)

## removePage

Removes a page from PageView.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCPageView.js:322</a>

#### Parameters

- `page` [Node](#)

### removePageAtIndex

Removes a page at index of PageView.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCPageView.js:338</a>

#### Parameters

- `index` [Number](#)

### removeAllPages

Removes all pages from PageView

meta	description
Defined in	<a href="#">cocos2d/core/components/CCPageView.js:354</a>

### scrollToPage

Scroll PageView to index.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCPageView.js:368</a>

#### Parameters

- `idx` [Number](#) index of page.

- `timeInSecond` [Number](#) scrolling time

### scrollToBottom

Scroll the content to the bottom boundary of ScrollView.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCScrollView.js:347</a>

### Parameters

- `timeInSecond` [Number](#) Scroll time in second, if you don't pass timeInSecond, the content will jump to the bottom boundary immediately.
- `attenuated` [Boolean](#) Whether the scroll acceleration attenuated, default is true.

### Examples

```
// Scroll to the bottom of the view.
scrollView.scrollToBottom(0.1);
```

### scrollToTop

Scroll the content to the top boundary of ScrollView.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCScrollView.js:372</a>

### Parameters

- `timeInSecond` [Number](#) Scroll time in second, if you don't pass timeInSecond, the content will jump to the top boundary immediately.
- `attenuated` [Boolean](#) Whether the scroll acceleration attenuated, default is true.

### Examples

```
// Scroll to the top of the view.
scrollView.scrollToTop(0.1);
```

### scrollToLeft

Scroll the content to the left boundary of ScrollView.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCScrollView.js:397</a>

## Parameters

- `timeInSecond` [Number](#) Scroll time in second, if you don't pass timeInSecond, the content will jump to the left boundary immediately.
- `attenuated` [Boolean](#) Whether the scroll acceleration attenuated, default is true.

## Examples

```
// Scroll to the left of the view.
scrollView.scrollToLeft(0.1);
```

### scrollToRight

Scroll the content to the right boundary of ScrollView.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCScrollView.js:422</a>

## Parameters

- `timeInSecond` [Number](#) Scroll time in second, if you don't pass timeInSecond, the content will jump to the right boundary immediately.
- `attenuated` [Boolean](#) Whether the scroll acceleration attenuated, default is true.

## Examples

```
// Scroll to the right of the view.
scrollView.scrollToRight(0.1);
```

### scrollToTopLeft

Scroll the content to the top left boundary of ScrollView.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCScrollView.js:447</a>

## Parameters

- `timeInSecond` [Number](#) Scroll time in second, if you don't pass `timeInSecond`, the content will jump to the top left boundary immediately.
- `attenuated` [Boolean](#) Whether the scroll acceleration attenuated, default is true.

## Examples

```
// Scroll to the upper left corner of the view.  
scrollView.scrollToTopLeft(0.1);
```

### scrollToTopRight

Scroll the content to the top right boundary of ScrollView.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCScrollView.js:472</a>

## Parameters

- `timeInSecond` [Number](#) Scroll time in second, if you don't pass `timeInSecond`, the content will jump to the top right boundary immediately.
- `attenuated` [Boolean](#) Whether the scroll acceleration attenuated, default is true.

## Examples

```
// Scroll to the top right corner of the view.  
scrollView.scrollToTopRight(0.1);
```

### scrollToBottomLeft

Scroll the content to the bottom left boundary of ScrollView.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCScrollView.js:497</a>

## Parameters

- `timeInSecond` [Number](#) Scroll time in second, if you don't pass `timeInSecond`, the content will jump to the bottom left boundary immediately.
- `attenuated` [Boolean](#) Whether the scroll acceleration attenuated, default is true.

## Examples

```
// Scroll to the lower left corner of the view.  
scrollView.scrollToBottomLeft(0.1);
```

### scrollToBottomRight

Scroll the content to the bottom right boundary of ScrollView.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCScrollView.js:522</a>

## Parameters

- `timeInSecond` [Number](#) Scroll time in second, if you don't pass timeInSecond, the content will jump to the bottom right boundary immediately.
- `attenuated` [Boolean](#) Whether the scroll acceleration attenuated, default is true.

## Examples

```
// Scroll to the lower right corner of the view.  
scrollView.scrollToBottomRight(0.1);
```

### scrollToOffset

Scroll with an offset related to the ScrollView's top left origin, if timeInSecond is omitted, then it will jump to the specific offset immediately.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCScrollView.js:548</a>

## Parameters

- `offset` [Vec2](#) A Vec2, the value of which each axis between 0 and maxScrollOffset
- `timeInSecond` [Number](#) Scroll time in second, if you don't pass timeInSecond, the content will jump to the specific offset of ScrollView immediately.
- `attenuated` [Boolean](#) Whether the scroll acceleration attenuated, default is true.

## Examples

```
// Scroll to middle position in 0.1 second in x-axis  
let maxScrollOffset = this.getMaxScrollOffset();  
scrollView.scrollToOffset(cc.v2(maxScrollOffset.x / 2, 0), 0.1);
```

## getScrollOffset

Get the positive offset value corresponds to the content's top left boundary.

meta	description
Returns	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/components/CCScrollView.js:582</a>

## getMaxScrollOffset

Get the maximize available scroll offset

meta	description
Returns	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/core/components/CCScrollView.js:595</a>

## scrollToPercentHorizontal

Scroll the content to the horizontal percent position of ScrollView.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCScrollView.js:612</a>

### Parameters

- percent [Number](#) A value between 0 and 1.
- timeInSecond [Number](#) Scroll time in second, if you don't pass timeInSecond, the content will jump to the horizontal percent position of ScrollView immediately.
- attenuated [Boolean](#) Whether the scroll acceleration attenuated, default is true.

### Examples

```
// Scroll to middle position.  
scrollView.scrollToBottomRight(0.5, 0.1);
```

## scrollTo

Scroll the content to the percent position of ScrollView in any direction.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCScrollView.js:638</a>

### Parameters

- **anchor** [Vec2](#) A point which will be clamp between cc.v2(0,0) and cc.v2(1,1).
- **timeInSecond** [Number](#) Scroll time in second, if you don't pass timeInSecond, the content will jump to the percent position of ScrollView immediately.
- **attenuated** [Boolean](#) Whether the scroll acceleration attenuated, default is true.

### Examples

```
// Vertical scroll to the bottom of the view.  
scrollView.scrollTo(cc.v2(0, 1), 0.1);  
  
// Horizontal scroll to view right.  
scrollView.scrollTo(cc.v2(1, 0), 0.1);
```

## scrollToPercentVertical

Scroll the content to the vertical percent position of ScrollView.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCScrollView.js:667</a>

### Parameters

- **percent** [Number](#) A value between 0 and 1.
- **timeInSecond** [Number](#) Scroll time in second, if you don't pass timeInSecond, the content will jump to the vertical percent position of ScrollView immediately.
- **attenuated** [Boolean](#) Whether the scroll acceleration attenuated, default is true. // Scroll to middle position. scrollView.scrollToPercentVertical(0.5, 0.1);

## stopAutoScroll

Stop auto scroll immediately

meta	description
Defined in	<a href="#">cocos2d/core/components/CCScrollView.js:692</a>

### setContentPosition

Modify the content position.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCScrollView.js:702</a>

### Parameters

- `position Vec2` The position in content's parent space.

### getContentPosition

Query the content's position in its parent space.

meta	description
Returns	Position
Defined in	<a href="#">cocos2d/core/components/CCScrollView.js:717</a>

### isScrolling

Query whether the user is currently dragging the ScrollView to scroll it

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCScrollView.js:727</a>

## isAutoScrolling

Query whether the ScrollView is currently scrolling because of a bounceback or inertia slowdown.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCScrollView.js:737</a>

## update

Update is called every frame, if the Component is enabled.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:222</a>

## Parameters

- dt [Number](#) the delta time in seconds it took to complete the last frame

## lateUpdate

LateUpdate is called every frame, if the Component is enabled.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:233</a>

## \_\_preload

\_\_preload is called before every onLoad. It is used to initialize the builtin components internally, to avoid checking whether onLoad is called before every public method calls. This method should be removed if script priority is supported.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:243</a>

## onLoad

When attaching to an active node or its node first activated. onLoad is always called before any start functions, this allows you to order initialization of scripts.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:254</a>

## start

Called before all scripts' update if the Component is enabled the first time. Usually used to initialize some logic which need to be called after all components' `onload` methods called.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:267</a>

## onEnable

Called when this component becomes enabled and its node is active.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:280</a>

## onDisable

Called when this component becomes disabled or its node becomes inactive.  
This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:290</a>

## onDestroy

Called when this component will be destroyed.  
This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:300</a>

## onFocusInEditor

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:310</a>

## onLostFocusInEditor

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:315</a>

## resetInEditor

Called to initialize the component or node's properties when adding the component the first time or when the Reset command is used. This function is only called in editor.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:320</a>

## addComponent

Adds a component class to the node. You can also add component to node by passing in the name of the script.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:330</a>

### Parameters

- typeOrClassName [Function](#) | [String](#) the constructor or the class name of the component to add

### Examples

```
var sprite = node.addComponent(cc.Sprite);
var test = node.addComponent("Test");
```

## getComponent

Returns the component of supplied type if the node has one attached, null if it doesn't.

You can also get component in the node by passing in the name of the script.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:348</a>

### Parameters

- typeOrClassName [Function](#) | [String](#)

## Examples

```
// get sprite component.  
var sprite = node.getComponent(cc.Sprite);  
// get custom test calss.  
var test = node.getComponent("Test");
```

## getComponents

Returns all components of supplied Type in the node.

meta	description
Returns	<a href="#">Component[]</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:372</a>

## Parameters

- typeOrClassName [Function](#) | [String](#)

## Examples

```
var sprites = node.getComponents(cc.Sprite);  
var tests = node.getComponents("Test");
```

## getComponentInChildren

Returns the component of supplied type in any of its children using depth first search.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:390</a>

## Parameters

- typeOrClassName [Function](#) | [String](#)

## Examples

```
var sprite = node.getComponentInChildren(cc.Sprite);  
var Test = node.getComponentInChildren("Test");
```

## getComponentsInChildren

Returns the components of supplied type in self or any of its children using depth first search.

meta	description
Returns	<a href="#">Component[]</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:408</a>

### Parameters

- `typeOrClassName` [Function](#) | [String](#)

### Examples

```
var sprites = node.getComponentsInChildren(cc.Sprite);
var tests = node.getComponentsInChildren("Test");
```

## \_getLocalBounds

If the component's bounding box is different from the node's, you can implement this method to supply a custom axis aligned bounding box (AABB), so the editor's scene view can perform hit test properly.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:428</a>

### Parameters

- `out_rect` [Rect](#) the Rect to receive the bounding box

## onRestore

onRestore is called after the user clicks the Reset item in the Inspector's context menu or performs an undo operation on this component.

If the component contains the "internal state", short for "temporary member variables which not included in its CCClass properties", then you may need to implement this function.

The editor will call the getset accessors of your component to record/restore the component's state for undo/redo operation. However, in extreme cases, it may not work well. Then you should implement this function to manually synchronize your component's "internal states" with its public properties. Once you implement this function, all the getset accessors of your component will not be called when the user performs an undo/redo operation. Which means that only the properties with default value will be recorded or restored by editor.

Similarly, the editor may fail to reset your component correctly in extreme cases. Then if you need to support the reset menu, you should manually synchronize your component's "internal states" with its properties in this function. Once you implement this function, all the getset accessors of your component will not be called during reset operation. Which means that only the properties with default value will be reset by editor.

This function is only called in editor mode.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:441</a>

### schedule

Schedules a custom selector. If the selector is already scheduled, then the interval parameter will be updated without scheduling it again.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:535</a>

### Parameters

- `callback` [function](#) The callback function
- `interval` [Number](#) Tick interval in seconds. 0 means tick every frame.
- `repeat` [Number](#) The selector will be executed (repeat + 1) times, you can use `cc.macro.REPEAT_FOREVER` for tick infinitely.

- `delay` [Number](#) The amount of time that the first tick will wait before execution.

### Examples

```
var timeCallback = function (dt) {
    cc.log("time: " + dt);
}
this.schedule(timeCallback, 1);
```

### scheduleOnce

Schedules a callback function that runs only once, with a delay of 0 or larger.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:572</a>

### Parameters

- `callback` [function](#) A function wrapped as a selector
- `delay` [Number](#) The amount of time that the first tick will wait before execution.

### Examples

```
var timeCallback = function (dt) {
    cc.log("time: " + dt);
}
this.scheduleOnce(timeCallback, 2);
```

### unschedule

Unschedules a custom callback function.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:589</a>

### Parameters

- `callback_fn` [function](#) A function wrapped as a selector

### Examples

```
this.unschedule(_callback);
```

## unscheduleAllCallbacks

unschedule all scheduled callback functions: custom callback functions, and the 'update' callback function.

Actions are not affected by this method.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:605</a>

## Examples

```
this.unscheduleAllCallbacks();
```

## destroy

Destroy this Object, and release all its own references to other objects.

Actual object destruction will delayed until before rendering. From the next frame, this object is not usable any more. You can use cc.isValid(obj) to check whether the object is destroyed before accessing it.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:296</a>

## Examples

```
obj.destroy();
```

## \_destruct

Clear all references in the instance.

NOTE: this method will not clear the getter or setter functions which defined in the instance of CCObject. You can override the \_destruct method if you need, for example:

```
_destruct: function () { for (var key in this) { if (this.hasOwnProperty(key)) { switch (typeof this[key]) { case 'string': this[key] = ""; break; case 'object': case 'function': this[key] = null; break; } } }
```

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:430</a>

### \_onPreDestroy

Called before the object being destroyed.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:463</a>

### \_serialize

The customized serialization for this object. (Editor Only)

meta	description
Returns	<a href="#">object</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:488</a>

### Parameters

- exporting [Boolean](#)

### \_deserialize

Init this object from the custom serialized data.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:498</a>

### Parameters

- data [Object](#) the serialized json data
- ctx [\\_Deserializer](#)

## *Events*

### **page-turning Event**

Module: [cc](#)

Note: This event is emitted from the node to which the component belongs.

## **Index**

### **Details**

#### **scroll-to-top Event**

Module: [cc](#)

Note: This event is emitted from the node to which the component belongs.

## **Index**

### **Details**

#### **scroll-to-bottom Event**

Module: [cc](#)

Note: This event is emitted from the node to which the component belongs.

## **Index**

### **Details**

#### **scroll-to-left Event**

Module: [cc](#)

Note: This event is emitted from the node to which the component belongs.

## **Index**

### **Details**

#### **scroll-to-right Event**

Module: [cc](#)

Note: This event is emitted from the node to which the component belongs.

## Index

### Details

#### **scrolling Event**

Module: [cc](#)

Note: This event is emitted from the node to which the component belongs.

## Index

### Details

#### **bounce-bottom Event**

Module: [cc](#)

Note: This event is emitted from the node to which the component belongs.

## Index

### Details

#### **bounce-top Event**

Module: [cc](#)

Note: This event is emitted from the node to which the component belongs.

## Index

### Details

#### **bounce-left Event**

Module: [cc](#)

Note: This event is emitted from the node to which the component belongs.

## Index

## Details

### bounce-right Event

Module: [cc](#)

Note: This event is emitted from the node to which the component belongs.

## Index

## Details

### scroll-ended Event

Module: [cc](#)

Note: This event is emitted from the node to which the component belongs.

## Index

## Details

### touch-up Event

Module: [cc](#)

Note: This event is emitted from the node to which the component belongs.

## Index

## Details

### scroll-began Event

Module: [cc](#)

Note: This event is emitted from the node to which the component belongs.

## PageViewIndicator Class

Extends [Component](#)

Module: [cc](#)

The Page View Indicator Component

## Index

## Properties

- `spriteFrame` SpriteFrame The spriteFrame for each element.
- `direction` PageViewIndicator.Direction The location direction of PageViewIndicator.
- `cellSize` Size The cellSize for each element.
- `spacing` Number The distance between each element.
- `_eventTargets` Array Register all related EventTargets,...
- `node` Node The node this component is attached to.
- `uuid` String The uuid for editor.
- `_enabled` Boolean
- `enabled` Boolean indicates whether this component is enabled or not.
- `enabledInHierarchy` Boolean indicates whether this component is enabled and its node is also active in the hierarchy.
- `_isOnLoadCalled` Number Returns a value which used to indicate the onLoad get called or not.
- `_name` String
- `_objFlags` Number
- `name` String The name of the object.
- `isValid` Boolean Indicates whether the object is not yet destroyed.

## Methods

- `setPageView` Set Page View
- `update` This is a lifecycle method.
- `lateUpdate` This is a lifecycle method.
- `__preload` \_\_preload is called before every onLoad.
- `onLoad` When attaching to an active node or its node first activated.
- `start` Called before all scripts' update if the Component is enabled the first time.
- `onEnable` This is a lifecycle method.
- `onDisable` This is a lifecycle method.
- `onDestroy` This is a lifecycle method.
- `onFocusInEditor`
- `onLostFocusInEditor`
- `resetInEditor` Called to initialize the component or node's properties when adding the component the first time or when the Reset command is used.
- `addComponent` Adds a component class to the node.
- `getComponent` Returns the component of supplied type if the node has one attached, null if it doesn't....
- `getComponents` Returns all components of supplied Type in the node.
- `getComponentInChildren` Returns the component of supplied type in any of its children using depth first search.
- `getComponentsInChildren` Returns the components of supplied type in self or any of its children using depth first search.
- `_getLocalBounds` If the component's bounding box is different from the node's, you can implement this method to supply
- `onRestore` for undo/redo operation.
- `schedule` Schedules a custom selector....
- `scheduleOnce` Schedules a callback function that runs only once, with a delay of 0 or larger.
- `unschedule` Unschedules a custom callback function.
- `unscheduleAllCallbacks` unschedule all scheduled callback functions: custom callback functions, and the 'update' callback function....
- `destroy` Actual object destruction will delayed until before rendering.
- `_destruct` Clear all references in the instance.

- `_onPreDestroy` Called before the object being destroyed.
- `_serialize` The customized serialization for this object.
- `_deserialize` Init this object from the custom serialized data.

## Details

### *Properties*

spriteFrame

The spriteFrame for each element.

meta	description
Type	<a href="#">SpriteFrame</a>
Defined in	<a href="#">cocos2d/core/components/CCPageViewIndicator.js:72</a>

direction

The location direction of PageViewIndicator.

meta	description
Type	<a href="#">PageViewIndicator.Direction</a>
Defined in	<a href="#">cocos2d/core/components/CCPageViewIndicator.js:83</a>

cellSize

The cellSize for each element.

meta	description
Type	<a href="#">Size</a>
Defined in	<a href="#">cocos2d/core/components/CCPageViewIndicator.js:94</a>

spacing

The distance between each element.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/CCPageViewIndicator.js:104</a>

### \_\_eventTargets

Register all related EventTargets, all event callbacks will be removed in `_onPreDestroy`

meta	description
Type	<a href="#">Array</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:61</a>

### node

The node this component is attached to. A component is always attached to a node.

meta	description
Type	<a href="#">Node</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:75</a>

### Examples

```
cc.log(comp.node);
```

### uuid

The uuid for editor.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:106</a>

## Examples

```
cc.log(comp.uuid);
```

### \_enabled

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:147</a>

### enabled

indicates whether this component is enabled or not.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:154</a>

## Examples

```
comp.enabled = true;  
cc.log(comp.enabled);
```

### enabledInHierarchy

indicates whether this component is enabled and its node is also active in the hierarchy.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:185</a>

## Examples

```
cc.log(comp.enabledInHierarchy);
```

## \_isOnLoadCalled

Returns a value which used to indicate the onLoad get called or not.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:201</a>

## Examples

```
cc.log(this._isOnLoadCalled > 0);
```

## \_name

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:76</a>

## \_objFlags

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:83</a>

## name

The name of the object.

meta	description
Type	<a href="#">String</a>

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:243</a>

## Examples

```
obj.name = "New Obj";
```

### isValid

Indicates whether the object is not yet destroyed. (It will not be available after being destroyed)  
When an object's `destroy` is called, it is actually destroyed after the end of this frame.  
So `isValid` will return false from the next frame, while `isValid` in the current frame will still be true.  
If you want to determine whether the current frame has called `destroy`, use `cc.isValid(obj, true)`, but this is often caused by a particular logical requirements, which is not normally required.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:261</a>

## Examples

```
var node = new cc.Node();
cc.log(node.isValid);      // true
node.destroy();
cc.log(node.isValid);      // true, still valid in this frame
// after a frame...
cc.log(node.isValid);      // false, destroyed in the end of last frame
```

## Methods

### setPageView

Set Page View

meta	description
Defined in	<a href="#">cocos2d/core/components/CCPageViewIndicator.js:123</a>

## Parameters

- target [PageView](#)

## update

Update is called every frame, if the Component is enabled.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:222</a>

## Parameters

- `dt` [Number](#) the delta time in seconds it took to complete the last frame

## lateUpdate

LateUpdate is called every frame, if the Component is enabled.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:233</a>

## \_\_preload

`__preload` is called before every `onLoad`. It is used to initialize the builtin components internally, to avoid checking whether `onLoad` is called before every public method calls. This method should be removed if script priority is supported.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:243</a>

## onLoad

When attaching to an active node or its node first activated. `onLoad` is always called before any start functions, this allows you to order initialization of scripts.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:254</a>

### start

Called before all scripts' update if the Component is enabled the first time. Usually used to initialize some logic which need to be called after all components' `onload` methods called. This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:267</a>

### onEnable

Called when this component becomes enabled and its node is active. This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:280</a>

### onDisable

Called when this component becomes disabled or its node becomes inactive. This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:290</a>

### onDestroy

Called when this component will be destroyed. This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

<b>meta</b>	<b>description</b>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:300</a>

#### onFocusInEditor

<b>meta</b>	<b>description</b>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:310</a>

#### onLostFocusInEditor

<b>meta</b>	<b>description</b>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:315</a>

#### resetInEditor

Called to initialize the component or node's properties when adding the component the first time or when the Reset command is used. This function is only called in editor.

<b>meta</b>	<b>description</b>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:320</a>

#### addComponent

Adds a component class to the node. You can also add component to node by passing in the name of the script.

<b>meta</b>	<b>description</b>
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:330</a>

## Parameters

- `typeOrClassName` [Function](#) | [String](#) the constructor or the class name of the component to add

## Examples

```
var sprite = node.addComponent(cc.Sprite);
var test = node.addComponent("Test");
```

## getComponent

Returns the component of supplied type if the node has one attached, null if it doesn't. You can also get component in the node by passing in the name of the script.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:348</a>

## Parameters

- `typeOrClassName` [Function](#) | [String](#)

## Examples

```
// get sprite component.
var sprite = node.getComponent(cc.Sprite);
// get custom test calss.
var test = node.getComponent("Test");
```

## getComponents

Returns all components of supplied Type in the node.

meta	description
Returns	<a href="#">Component[]</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:372</a>

## Parameters

- `typeOrClassName` [Function](#) | [String](#)

## Examples

```
var sprites = node.getComponents(cc.Sprite);
var tests = node.getComponents("Test");
```

### getComponentInChildren

Returns the component of supplied type in any of its children using depth first search.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:390</a>

## Parameters

- `typeOrClassName` [Function](#) | [String](#)

## Examples

```
var sprite = node.getComponentInChildren(cc.Sprite);
var Test = node.getComponentInChildren("Test");
```

### getComponentsInChildren

Returns the components of supplied type in self or any of its children using depth first search.

meta	description
Returns	<a href="#">Component[]</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:408</a>

## Parameters

- `typeOrClassName` [Function](#) | [String](#)

## Examples

```
var sprites = node.getComponentsInChildren(cc.Sprite);
var tests = node.getComponentsInChildren("Test");
```

## `_getLocalBounds`

If the component's bounding box is different from the node's, you can implement this method to supply a custom axis aligned bounding box (AABB), so the editor's scene view can perform hit test properly.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:428</a>

### Parameters

- `out_rect Rect` the Rect to receive the bounding box

### onRestore

`onRestore` is called after the user clicks the Reset item in the Inspector's context menu or performs an undo operation on this component.

If the component contains the "internal state", short for "temporary member variables which not included in its CCClass properties", then you may need to implement this function.

The editor will call the getset accessors of your component to record/restore the component's state for undo/redo operation. However, in extreme cases, it may not work well. Then you should implement this function to manually synchronize your component's "internal states" with its public properties. Once you implement this function, all the getset accessors of your component will not be called when the user performs an undo/redo operation. Which means that only the properties with default value will be recorded or restored by editor.

Similarly, the editor may fail to reset your component correctly in extreme cases. Then if you need to support the reset menu, you should manually synchronize your component's "internal states" with its properties in this function. Once you implement this function, all the getset accessors of your component will not be called during reset operation. Which means that only the properties with default value will be reset by editor.

This function is only called in editor mode.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:441</a>

## schedule

Schedules a custom selector.

If the selector is already scheduled, then the interval parameter will be updated without scheduling it again.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:535</a>

### Parameters

- `callback` [function](#) The callback function
- `interval` [Number](#) Tick interval in seconds. 0 means tick every frame.
- `repeat` [Number](#) The selector will be executed (repeat + 1) times, you can use cc.macro.REPEAT\_FOREVER for tick infinitely.
- `delay` [Number](#) The amount of time that the first tick will wait before execution.

### Examples

```
var timeCallback = function (dt) {
    cc.log("time: " + dt);
}
this.schedule(timeCallback, 1);
```

## scheduleOnce

Schedules a callback function that runs only once, with a delay of 0 or larger.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:572</a>

### Parameters

- `callback` [function](#) A function wrapped as a selector
- `delay` [Number](#) The amount of time that the first tick will wait before execution.

## Examples

```
var timeCallback = function (dt) {  
    cc.log("time: " + dt);  
}  
this.scheduleOnce(timeCallback, 2);
```

## unschedule

Unschedules a custom callback function.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:589</a>

## Parameters

- `callback_fn` [function](#) A function wrapped as a selector

## Examples

```
this.unschedule(_callback);
```

## unscheduleAllCallbacks

unschedule all scheduled callback functions: custom callback functions, and the 'update' callback function.

Actions are not affected by this method.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:605</a>

## Examples

```
this.unscheduleAllCallbacks();
```

## destroy

Destroy this Object, and release all its own references to other objects.

Actual object destruction will delayed until before rendering. From the next frame, this object is not usable any more. You can use `cc.isValid(obj)` to check whether the object is destroyed before accessing it.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:296</a>

## Examples

```
obj.destroy();
```

### \_destruct

Clear all references in the instance.

NOTE: this method will not clear the getter or setter functions which defined in the instance of CCObject. You can override the \_destruct method if you need, for example:

```
_destruct: function () { for (var key in this) { if (this.hasOwnProperty(key)) { switch (typeof this[key]) { case 'string': this[key] = ''; break; case 'object': case 'function': this[key] = null; break; } } }}
```

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:430</a>

### \_onPreDestroy

Called before the object being destroyed.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:463</a>

### \_serialize

The customized serialization for this object. (Editor Only)

meta	description
Returns	<a href="#">object</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:488</a>

## Parameters

- `exporting Boolean`

`_deserialize`

Init this object from the custom serialized data.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:498</a>

## Parameters

- `data Object` the serialized json data
- `ctx _Deserializer`

# ParticleAsset Class

Extends [Asset](#)

Module: [cc](#)

Class for particle asset handling.

## Index

### Properties

- `loaded Boolean` Whether the asset is loaded or not
- `nativeUrl String` Returns the url of this asset's native object, if none it will returns an empty string.
- `_native String` Serializable url for native asset.
- `_nativeAsset Object` The underlying native asset of this asset if one is available.
- `_uuid String`
- `_name String`
- `_objFlags Number`
- `name String` The name of the object.
- `isValid Boolean` Indicates whether the object is not yet destroyed.

### Methods

- `toString` Returns the asset's url.
- `serialize` 应 AssetDB 要求提供这个方法
- `createNode` Create a new node using this asset in the scene....

- `_setRawAsset` Set native file name for this asset.
- `destroy` Actual object destruction will delayed until before rendering.
- `_destruct` Clear all references in the instance.
- `_onPreDestroy` Called before the object being destroyed.
- `_serialize` The customized serialization for this object.
- `_deserialize` Init this object from the custom serialized data.

## Details

### *Properties*

loaded

Whether the asset is loaded or not

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:57</a>

nativeUrl

Returns the url of this asset's native object, if none it will returns an empty string.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:70</a>

\_native

Serializable url for native asset.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:108</a>

### \_nativeAsset

The underlying native asset of this asset if one is available. This property can be used to access additional details or functionality related to the asset. This property will be initialized by the loader if `_native` is available.

<b>meta</b>	<b>description</b>
Type	<a href="#">Object</a>
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:116</a>

### \_uuid

<b>meta</b>	<b>description</b>
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/assets/CCRawAsset.js:46</a>

### \_name

<b>meta</b>	<b>description</b>
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:76</a>

### \_objFlags

<b>meta</b>	<b>description</b>
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:83</a>

### name

The name of the object.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:243</a>

## Examples

```
obj.name = "New Obj";
```

### isValid

Indicates whether the object is not yet destroyed. (It will not be available after being destroyed)  
When an object's `destroy` is called, it is actually destroyed after the end of this frame.  
So `isValid` will return false from the next frame, while `isValid` in the current frame will still be true.  
If you want to determine whether the current frame has called `destroy`, use `cc.isValid(obj, true)`, but this is often caused by a particular logical requirements, which is not normally required.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:261</a>

## Examples

```
var node = new cc.Node();
cc.log(node.isValid);      // true
node.destroy();
cc.log(node.isValid);      // true, still valid in this frame
// after a frame...
cc.log(node.isValid);      // false, destroyed in the end of last frame
```

## Methods

### toString

Returns the asset's url.

The Asset object overrides the `toString()` method of the Object object. For Asset objects, the `toString()` method returns a string representation of the object. JavaScript calls the `toString()` method automatically when an asset is to be represented as a text value or when a texture is referred to in a string concatenation.

meta	description
Returns	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:165</a>

serialize

应 AssetDB 要求提供这个方法

meta	description
Returns	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:179</a>

createNode

Create a new node using this asset in the scene.

If this type of asset dont have its corresponding node type, this method should be null.

meta	description
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:190</a>

Parameters

- callback [Function](#)
- error [String](#) null or the error info
- node [Object](#) the created node or null

\_setRawAsset

Set native file name for this asset.

meta	description
Defined in	<a href="#">cocos2d/core/assets/CCAsset.js:205</a>

## Parameters

- `filename` [String](#)
- `inLibrary` [Boolean](#)

## destroy

Destroy this Object, and release all its own references to other objects.

Actual object destruction will delayed until before rendering. From the next frame, this object is not usable any more. You can use `cc.isValid(obj)` to check whether the object is destroyed before accessing it.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:296</a>

## Examples

```
obj.destroy();
```

```
_destruct
```

Clear all references in the instance.

NOTE: this method will not clear the getter or setter functions which defined in the instance of CCObject. You can override the `_destruct` method if you need, for example:

```
_destruct: function () { for (var key in this) { if (this.hasOwnProperty(key)) { switch (typeof this[key]) { case 'string': this[key] = ""; break; case 'object': case 'function': this[key] = null; break; } } }}
```

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:430</a>

```
_onPreDestroy
```

Called before the object being destroyed.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:463</a>

### \_serialize

The customized serialization for this object. (Editor Only)

meta	description
Returns	<a href="#">object</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:488</a>

Parameters

- exporting [Boolean](#)

### \_deserialize

Init this object from the custom serialized data.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:498</a>

Parameters

- data [Object](#) the serialized json data
- ctx [\\_Deserializer](#)

## ParticleSystem Class

Extends [Component](#)

Module: [cc](#)

Particle System base class.

Attributes of a Particle System:

- emmision rate of the particles
- Gravity Mode (Mode A):
- gravity
- direction
- speed +- variance
- tangential acceleration +- variance
- radial acceleration +- variance
- Radius Mode (Mode B):

- startRadius +- variance
- endRadius +- variance
- rotate +- variance
- Properties common to all modes:
- life +- life variance
- start spin +- variance
- end spin +- variance
- start size +- variance
- end size +- variance
- start color +- variance
- end color +- variance
- life +- variance
- blending function
- texture

cocos2d also supports particles generated by Particle Designer  
[\(http://particledesigner.71squared.com/\)](http://particledesigner.71squared.com/).

'Radius Mode' in Particle Designer uses a fixed emit rate of 30 hz. Since that can't be guaranteed in cocos2d,

cocos2d uses another approach, but the results are almost identical.

cocos2d supports all the variables used by Particle Designer plus a bit more:

- spinning particles (supported when using ParticleSystem)
  - tangential acceleration (Gravity mode)
  - radial acceleration (Gravity mode)
  - radius direction (Radius mode) (Particle Designer supports outwards to inwards direction only)
- It is possible to customize any of the above mentioned properties in runtime. Example:

## Index

### Properties

- `preview` Boolean Play particle in edit mode.
- `custom` Boolean If set custom to true, then use custom properties instead of read particle file.
- `file` string The plist file.
- `spriteFrame` SpriteFrame SpriteFrame used for particles display
- `texture` String Texture of Particle System, readonly, please use spriteFrame to setup new texture.
- `particleCount` Number Current quantity of particles that are being simulated.
- `stopped` Boolean Indicate whether the system simulation have stopped.
- `playOnLoad` Boolean If set to true, the particle system will automatically start playing on onLoad.
- `autoRemoveOnFinish` Boolean Indicate whether the owner node will be auto-removed when it has no particles left.
- `active` Boolean Indicate whether the particle system is activated.
- `totalParticles` Number Maximum particles of the system.
- `duration` Number How many seconds the emitter will run.
- `emissionRate` Number Emission rate of the particles.
- `life` Number Life of each particle setter.
- `lifeVar` Number Variation of life.
- `startColor` cc.Color Start color of each particle.
- `startColorVar` cc.Color Variation of the start color.
- `endColor` cc.Color Ending color of each particle.

- `endColorVar` `cc.Color` Variation of the end color.
- `angle` `Number` Angle of each particle setter.
- `angleVar` `Number` Variation of angle of each particle setter.
- `startSize` `Number` Start size in pixels of each particle.
- `startSizeVar` `Number` Variation of start size in pixels.
- `endSize` `Number` End size in pixels of each particle.
- `endSizeVar` `Number` Variation of end size in pixels.
- `startSpin` `Number` Start angle of each particle.
- `startSpinVar` `Number` Variation of start angle.
- `endSpin` `Number` End angle of each particle.
- `endSpinVar` `Number` Variation of end angle.
- `sourcePos` `Vec2` Source position of the emitter.
- `posVar` `Vec2` Variation of source position.
- `positionType` `ParticleSystem.PositionType` Particles movement type.
- `emitterMode` `ParticleSystemEmitterMode` Particles emitter modes.
- `gravity` `Vec2` Gravity of the emitter.
- `speed` `Number` Speed of the emitter.
- `speedVar` `Number` Variation of the speed.
- `tangentialAccel` `Number` Tangential acceleration of each particle.
- `tangentialAccelVar` `Number` Variation of the tangential acceleration.
- `radialAccel` `Number` Acceleration of each particle.
- `radialAccelVar` `Number` Variation of the radial acceleration.
- `rotationIsDir` `Boolean` Indicate whether the rotation of each particle equals to its direction.
- `startRadius` `Number` Starting radius of the particles.
- `startRadiusVar` `Number` Variation of the starting radius.
- `endRadius` `Number` Ending radius of the particles.
- `endRadiusVar` `Number` Variation of the ending radius.
- `rotatePerS` `Number` Number of degress to rotate a particle around the source pos per second.
- `rotatePerSVar` `Number` Variation of the degress to rotate a particle around the source pos per second.
- `DURATION_INFINITY` `Number` The Particle emitter lives forever.
- `START_SIZE_EQUAL_TO_END_SIZE` `Number` The starting size of the particle is equal to the ending size.
- `START_RADIUS_EQUAL_TO_END_RADIUS` `Number` The starting radius of the particle is equal to the ending radius.
- `__eventTargets` `Array` Register all related EventTargets,...
- `node` `Node` The node this component is attached to.
- `uuid` `String` The uuid for editor.
- `_enabled` `Boolean`
- `enabled` `Boolean` indicates whether this component is enabled or not.
- `enabledInHierarchy` `Boolean` indicates whether this component is enabled and its node is also active in the hierarchy.
- `_isOnLoadCalled` `Number` Returns a value which used to indicate the onLoad get called or not.
- `_name` `String`
- `_objFlags` `Number`
- `name` `String` The name of the object.
- `isValid` `Boolean` Indicates whether the object is not yet destroyed.

## Methods

- `stopSystem` Stop emitting particles.
- `resetSystem` Kill all living particles.
- `isFull` Whether or not the system is full.

- `setTextureWithRect` Sets a new texture with a rect.
- `update` This is a lifecycle method.
- `lateUpdate` This is a lifecycle method.
- `__preload` `__preload` is called before every `onLoad`.
- `onLoad` When attaching to an active node or its node first activated.
- `start` Called before all scripts' update if the Component is enabled the first time.
- `onEnable` This is a lifecycle method.
- `onDisable` This is a lifecycle method.
- `onDestroy` This is a lifecycle method.
- `onFocusInEditor`
- `onLostFocusInEditor`
- `resetInEditor` Called to initialize the component or node's properties when adding the component the first time or when the Reset command is used.
- `addComponent` Adds a component class to the node.
- `getComponent` Returns the component of supplied type if the node has one attached, null if it doesn't....
- `getComponents` Returns all components of supplied Type in the node.
- `getComponentInChildren` Returns the component of supplied type in any of its children using depth first search.
- `getComponentsInChildren` Returns the components of supplied type in self or any of its children using depth first search.
- `_getLocalBounds` If the component's bounding box is different from the node's, you can implement this method to supply
- `onRestore` for undo/redo operation.
- `schedule` Schedules a custom selector....
- `scheduleOnce` Schedules a callback function that runs only once, with a delay of 0 or larger.
- `unschedule` Unschedules a custom callback function.
- `unscheduleAllCallbacks` unschedule all scheduled callback functions: custom callback functions, and the 'update' callback function....
- `destroy` Actual object destruction will delayed until before rendering.
- `_destruct` Clear all references in the instance.
- `_onPreDestroy` Called before the object being destroyed.
- `_serialize` The customized serialization for this object.
- `_deserialize` Init this object from the custom serialized data.

## Details

### Properties

`preview`

Play particle in edit mode.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:123</a>

## custom

If set custom to true, then use custom properties instead of read particle file.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:144</a>

## file

The plist file.

meta	description
Type	<a href="#">string</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:175</a>

## spriteFrame

SpriteFrame used for particles display

meta	description
Type	<a href="#">SpriteFrame</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:209</a>

## texture

Texture of Particle System, readonly, please use spriteFrame to setup new texture.

meta	description
Type	<a href="#">String</a>

meta	description
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:248</a>

### particleCount

Current quantity of particles that are being simulated.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:269</a>

### stopped

Indicate whether the system simulation have stopped.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:283</a>

### playOnLoad

If set to true, the particle system will automatically start playing on onLoad.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:297</a>

### autoRemoveOnFinish

Indicate whether the owner node will be auto-removed when it has no particles left.

<b>meta</b>	<b>description</b>
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:306</a>

### active

Indicate whether the particle system is activated.

<b>meta</b>	<b>description</b>
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:317</a>

### totalParticles

Maximum particles of the system.

<b>meta</b>	<b>description</b>
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:330</a>

### duration

How many seconds the emitter wil run. -1 means 'forever'.

<b>meta</b>	<b>description</b>
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:337</a>

## **emissionRate**

Emission rate of the particles.

<b>meta</b>	<b>description</b>
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:344</a>

## **life**

Life of each particle setter.

<b>meta</b>	<b>description</b>
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:351</a>

## **lifeVar**

Variation of life.

<b>meta</b>	<b>description</b>
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:358</a>

## **startColor**

Start color of each particle.

<b>meta</b>	<b>description</b>
Type	cc.Color
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:366</a>

## startColorVar

Variation of the start color.

meta	description
Type	cc.Color
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:385</a>

## endColor

Ending color of each particle.

meta	description
Type	cc.Color
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:404</a>

## endColorVar

Variation of the end color.

meta	description
Type	cc.Color
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:423</a>

## angle

Angle of each particle setter.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:443</a>

## angleVar

Variation of angle of each particle setter.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:450</a>

## startSize

Start size in pixels of each particle.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:457</a>

## startSizeVar

Variation of start size in pixels.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:464</a>

## endSize

End size in pixels of each particle.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:471</a>

## endSizeVar

Variation of end size in pixels.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:478</a>

## startSpin

Start angle of each particle.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:485</a>

## startSpinVar

Variation of start angle.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:492</a>

## endSpin

End angle of each particle.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:499</a>

## endSpinVar

Variation of end angle.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:506</a>

## sourcePos

Source position of the emitter.

meta	description
Type	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:514</a>

## posVar

Variation of source position.

meta	description
Type	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:522</a>

## positionType

Particles movement type.

meta	description
Type	<a href="#">ParticleSystem.PositionType</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:530</a>

## emitterMode

Particles emitter modes.

meta	description
Type	<a href="#">ParticleSystemEmitterMode</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:541</a>

## gravity

Gravity of the emitter.

meta	description
Type	<a href="#">Vec2</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:554</a>

## speed

Speed of the emitter.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:561</a>

## speedVar

Variation of the speed.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:568</a>

## tangentialAccel

Tangential acceleration of each particle. Only available in 'Gravity' mode.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:575</a>

## tangentialAccelVar

Variation of the tangential acceleration.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:582</a>

## radialAccel

Acceleration of each particle. Only available in 'Gravity' mode.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:589</a>

## radialAccelVar

Variation of the radial acceleration.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:596</a>

## rotationIsDir

Indicate whether the rotation of each particle equals to its direction. Only available in 'Gravity' mode.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:604</a>

## startRadius

Starting radius of the particles. Only available in 'Radius' mode.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:614</a>

## startRadiusVar

Variation of the starting radius.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:621</a>

## endRadius

Ending radius of the particles. Only available in 'Radius' mode.

meta	description
Type	<a href="#">Number</a>

<b>meta</b>	<b>description</b>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:628</a>

### endRadiusVar

Variation of the ending radius.

<b>meta</b>	<b>description</b>
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:635</a>

### rotatePerS

Number of degress to rotate a particle around the source pos per second. Only available in 'Radius' mode.

<b>meta</b>	<b>description</b>
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:642</a>

### rotatePerSVar

Variation of the degress to rotate a particle around the source pos per second.

<b>meta</b>	<b>description</b>
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:649</a>

### DURATION\_INFINITY

The Particle emitter lives forever.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:730</a>

### START\_SIZE\_EQUAL\_TO\_END\_SIZE

The starting size of the particle is equal to the ending size.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:740</a>

### START\_RADIUS\_EQUAL\_TO\_END\_RADIUS

The starting radius of the particle is equal to the ending radius.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:750</a>

### \_\_eventTargets

Register all related EventTargets, all event callbacks will be removed in `_onPreDestroy`

meta	description
Type	<a href="#">Array</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:61</a>

## node

The node this component is attached to. A component is always attached to a node.

meta	description
Type	<a href="#">Node</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:75</a>

## Examples

```
cc.log(comp.node);
```

## uuid

The uuid for editor.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:106</a>

## Examples

```
cc.log(comp.uuid);
```

## \_enabled

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:147</a>

## enabled

indicates whether this component is enabled or not.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:154</a>

## Examples

```
comp.enabled = true;
cc.log(comp.enabled);
```

## enabledInHierarchy

indicates whether this component is enabled and its node is also active in the hierarchy.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:185</a>

## Examples

```
cc.log(comp.enabledInHierarchy);
```

## \_isOnLoadCalled

Returns a value which used to indicate the onLoad get called or not.

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:201</a>

## Examples

```
cc.log(this._isOnLoadCalled > 0);
```

### `_name`

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:76</a>

### `_objFlags`

meta	description
Type	<a href="#">Number</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:83</a>

### `name`

The name of the object.

meta	description
Type	<a href="#">String</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:243</a>

### Examples

```
obj.name = "New Obj";
```

### `isValid`

Indicates whether the object is not yet destroyed. (It will not be available after being destroyed)  
When an object's `destroy` is called, it is actually destroyed after the end of this frame.  
So `isValid` will return false from the next frame, while `isValid` in the current frame will still be true.  
If you want to determine whether the current frame has called `destroy`, use `cc.isValid(obj, true)`, but this is often caused by a particular logical requirements, which is not normally required.

meta	description
Type	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:261</a>

## Examples

```
var node = new cc.Node();
cc.log(node.isValid);      // true
node.destroy();
cc.log(node.isValid);      // true, still valid in this frame
// after a frame...
cc.log(node.isValid);      // false, destroyed in the end of last frame
```

## Methods

### stopSystem

Stop emitting particles. Running particles will continue to run until they die.

meta	description
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:870</a>

## Examples

```
// stop particle system.
myParticleSystem.stopSystem();
```

### resetSystem

Kill all living particles.

meta	description
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:883</a>

## Examples

```
// play particle system.
myParticleSystem.resetSystem();
```

## isFull

Whether or not the system is full.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:902</a>

## setTextureWithRect

Sets a new texture with a rect. The rect is in texture position and size. Please use spriteFrame property instead, this function is deprecated since v1.9

meta	description
Defined in	<a href="#">cocos2d/particle/CCParticleSystem.js:912</a>
Deprecated	since v1.9

## Parameters

- `texture Texture2D`
- `rect Rect`

## update

Update is called every frame, if the Component is enabled.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:222</a>

## Parameters

- `dt Number the delta time in seconds it took to complete the last frame`

## lateUpdate

LateUpdate is called every frame, if the Component is enabled.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:233</a>

## \_\_preload

\_\_preload is called before every onLoad. It is used to initialize the builtin components internally, to avoid checking whether onLoad is called before every public method calls. This method should be removed if script priority is supported.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:243</a>

## onLoad

When attaching to an active node or its node first activated. onLoad is always called before any start functions, this allows you to order initialization of scripts.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:254</a>

## start

Called before all scripts' update if the Component is enabled the first time. Usually used to initialize some logic which need to be called after all components' onload methods called.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:267</a>

## onEnable

Called when this component becomes enabled and its node is active.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:280</a>

## onDisable

Called when this component becomes disabled or its node becomes inactive.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:290</a>

## onDestroy

Called when this component will be destroyed.

This is a lifecycle method. It may not be implemented in the super class. You can only call its super class method inside it. It should not be called manually elsewhere.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:300</a>

## onFocusInEditor

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:310</a>

## onLostFocusInEditor

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:315</a>

## resetInEditor

Called to initialize the component or node's properties when adding the component the first time or when the Reset command is used. This function is only called in editor.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:320</a>

## addComponent

Adds a component class to the node. You can also add component to node by passing in the name of the script.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:330</a>

### Parameters

- typeOrClassName [Function](#) | [String](#) the constructor or the class name of the component to add

### Examples

```
var sprite = node.addComponent(cc.Sprite);
var test = node.addComponent("Test");
```

## getComponent

Returns the component of supplied type if the node has one attached, null if it doesn't. You can also get component in the node by passing in the name of the script.

meta	description
Returns	<a href="#">Component</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:348</a>

#### Parameters

- typeOrClassName [Function](#) | [String](#)

#### Examples

```
// get sprite component.
var sprite = node.getComponent(cc.Sprite);
// get custom test calss.
var test = node.getComponent("Test");
```

### getComponents

Returns all components of supplied Type in the node.

meta	description
Returns	<a href="#">Component[]</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:372</a>

#### Parameters

- typeOrClassName [Function](#) | [String](#)

#### Examples

```
var sprites = node.getComponents(cc.Sprite);
var tests = node.getComponents("Test");
```

### getComponentInChildren

Returns the component of supplied type in any of its children using depth first search.

meta	description
Returns	<a href="#">Component</a>

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:390</a>

## Parameters

- typeOrClassName [Function](#) | [String](#)

## Examples

```
var sprite = node.getComponentInChildren(cc.Sprite);
var Test = node.getComponentInChildren("Test");
```

## getComponentsInChildren

Returns the components of supplied type in self or any of its children using depth first search.

meta	description
Returns	<a href="#">Component[]</a>
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:408</a>

## Parameters

- typeOrClassName [Function](#) | [String](#)

## Examples

```
var sprites = node.getComponentsInChildren(cc.Sprite);
var tests = node.getComponentsInChildren("Test");
```

## \_getLocalBounds

If the component's bounding box is different from the node's, you can implement this method to supply a custom axis aligned bounding box (AABB), so the editor's scene view can perform hit test properly.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:428</a>

## Parameters

- `out_rect Rect` the Rect to receive the bounding box

### onRestore

onRestore is called after the user clicks the Reset item in the Inspector's context menu or performs an undo operation on this component.

If the component contains the "internal state", short for "temporary member variables which not included in its CCClass properties", then you may need to implement this function.

The editor will call the getset accessors of your component to record/restore the component's state for undo/redo operation. However, in extreme cases, it may not work well. Then you should implement this function to manually synchronize your component's "internal states" with its public properties. Once you implement this function, all the getset accessors of your component will not be called when the user performs an undo/redo operation. Which means that only the properties with default value will be recorded or restored by editor.

Similarly, the editor may fail to reset your component correctly in extreme cases. Then if you need to support the reset menu, you should manually synchronize your component's "internal states" with its properties in this function. Once you implement this function, all the getset accessors of your component will not be called during reset operation. Which means that only the properties with default value will be reset by editor.

This function is only called in editor mode.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:441</a>

### schedule

Schedules a custom selector.

If the selector is already scheduled, then the interval parameter will be updated without scheduling it again.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:535</a>

## Parameters

- `callback` [function](#) The callback function
- `interval` [Number](#) Tick interval in seconds. 0 means tick every frame.
- `repeat` [Number](#) The selector will be executed (repeat + 1) times, you can use cc.macro.REPEAT\_FOREVER for tick infinitely.
- `delay` [Number](#) The amount of time that the first tick will wait before execution.

## Examples

```
var timeCallback = function (dt) {
    cc.log("time: " + dt);
}
this.schedule(timeCallback, 1);
```

## scheduleOnce

Schedules a callback function that runs only once, with a delay of 0 or larger.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:572</a>

## Parameters

- `callback` [function](#) A function wrapped as a selector
- `delay` [Number](#) The amount of time that the first tick will wait before execution.

## Examples

```
var timeCallback = function (dt) {
    cc.log("time: " + dt);
}
this.scheduleOnce(timeCallback, 2);
```

## unschedule

Unschedules a custom callback function.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:589</a>

## Parameters

- `callback_fn` [function](#) A function wrapped as a selector

## Examples

```
this.unschedule(_callback);
```

### unscheduleAllCallbacks

unschedule all scheduled callback functions: custom callback functions, and the 'update' callback function.

Actions are not affected by this method.

meta	description
Defined in	<a href="#">cocos2d/core/components/CCComponent.js:605</a>

## Examples

```
this.unscheduleAllCallbacks();
```

### destroy

Destroy this Object, and release all its own references to other objects.

Actual object destruction will delayed until before rendering. From the next frame, this object is not usable any more. You can use `cc.isValid(obj)` to check whether the object is destroyed before accessing it.

meta	description
Returns	<a href="#">Boolean</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:296</a>

## Examples

```
obj.destroy();
```

### \_destruct

Clear all references in the instance.

NOTE: this method will not clear the getter or setter functions which defined in the instance of CCObject. You can override the \_destruct method if you need, for example: \_destruct: function () { for (var key in this) { if (this.hasOwnProperty(key)) { switch (typeof this[key]) { case 'string': this[key] = ""; break; case 'object': case 'function': this[key] = null; break; } } }

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:430</a>

### \_onPreDestroy

Called before the object being destroyed.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:463</a>

### \_serialize

The customized serialization for this object. (Editor Only)

meta	description
Returns	<a href="#">object</a>
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:488</a>

### Parameters

- exporting [Boolean](#)

### \_deserialize

Init this object from the custom serialized data.

meta	description
Defined in	<a href="#">cocos2d/core/platform/CCObject.js:498</a>

#### Parameters

- `data Object` the serialized json data
- `ctx _Deserializer`

## Editor

### Static Methods

#### Editor.KeyCode (key)

- `key number|string` - Can be number(key-code) or string(key-name)

Convenience method returns corresponding value for given keyName or keyCode.

### Static Properties

#### Editor.isDarwin

Indicates whether executes in OSX.

#### Editor.isElectron

Indicates whether executes in electron.

#### Editor.isMainProcess

Indicates whether executes in editor's main process.

#### Editor.isNative

Indicates whether executes in native environment (compare to web-browser).

## **Editor.isNode**

Indicates whether executes in node.js application.

## **Editor.isPureWeb**

Indicates whether executes in common web browser.

## **Editor.isRendererProcess**

Indicates whether executes in editor's core process (electron's browser context).

## **Editor.isRetina**

Check if running in retina display.

## **Editor.isWin32**

Indicates whether executes in Windows.

## **Editor.Easing**

### **Methods**

**Editor.Easing.linear (k)**

**Editor.Easing.quadIn (k)**

**Editor.Easing.quadOut (k)**

**Editor.Easing.quadInOut (k)**

**Editor.Easing.quadOutIn (k)**

**Editor.Easing.cubicIn (k)**

**Editor.Easing.cubicOut (k)**

**Editor.Easing.cubicInOut (k)**

**Editor.Easing.cubicOutIn (k)**

**Editor.Easing.quartIn (k)**

**Editor.Easing.quartOut (k)**

**Editor.Easing.quartInOut (k)**

**Editor.Easing.quartOutIn (k)**

**Editor.Easing.quintIn (k)**

**Editor.Easing.quintOut (k)**

**Editor.Easing.quintInOut (k)**

**Editor.Easing.quintOutIn (k)**

**Editor.Easing.sineln (k)**

**Editor.Easing.sineOut (k)**

**Editor.Easing.sinelnOut (k)**

**Editor.Easing.sineOutIn (k)**

**Editor.Easing.expoin (k)**

**Editor.Easing.expoOut (k)**

**Editor.Easing.expoinOut (k)**

**Editor.Easing.expoOutIn (k)**

**Editor.Easing.circIn (k)**

**Editor.Easing.circOut (k)**

**Editor.Easing.circInOut (k)**

**Editor.Easing.circOutIn (k)**

**Editor.Easing.elasticIn (k)**

**Editor.Easing.elasticOut (k)**

**Editor.Easing.elasticInOut (k)**

**Editor.Easing.elasticOutIn (k)**

**Editor.Easing.backIn (k)**

**Editor.Easing.backOut (k)**

**Editor.Easing.backInOut (k)**

**Editor.Easing.backOutIn (k)**

**Editor.Easing.bounceIn (k)**

**Editor.Easing.bounceOut (k)**

**Editor.Easing.bounceInOut (k)**

**Editor.Easing.bounceOutIn (k)**

**Editor.Easing.fade (k)**

## **Editor.IpcListener**

**Class: Editor.IpcListener**

**new Editor.IpcListener ()**

## Instance Methods

### ipcListener.on (message, callback)

- `message` string
- `callback` function

Register IPC message and respond it with the callback function.

### ipcListener.once (message, callback)

- `message` string
- `callback` function

Register IPC message and respond it once with the callback function.

### ipcListener.clear ()

Clear all registered IPC messages in the listener.

## Editor.JS

Extending JavaScript to better handle property and class inheritance.

## Methods

### Editor.JS.addon (obj, ...args)

- `obj` object
- `...args` object

Copy all properties not defined in `obj` from `arguments[1...n]` to it.

### Editor.JS.assign (obj, ...args)

- `obj` object
- `...args` object

Copy all properties from `arguments[1...n]` to `obj`, return the mixed result.

### Editor.JS.assignExcept (obj, src, except)

- `obj` object
- `src` object
- `except` array

Copy all properties from arguments[1...n] to `obj` except the specific ones.

## **Editor.JS.clear (obj)**

- `obj` object

Removes all enumerable properties from object.

## **Editor.JS.copyprop (name, source, target)**

- `name` string
- `source` object
- `target` object

Copy property by name from source to target.

## **Editor.JS.extend (cls, base)**

- `cls` function
- `base` function

Derive the class from the supplied base class.

## **Editor.JS.extract (obj, propNames)**

- `obj` object
- `except` array(string)

Extract properties by `propNames` from `obj`, return the extracted result.

## **Editor.JS.getPropertyByPath (obj, path)**

- `obj` object
- `path` string

Get property by path.

# **Editor.Math**

# Properties

**Editor.Math.EPSILON**

**Editor.Math.MACHINE\_EPSILON**

**Editor.Math.TWO\_PI**

**Editor.Math.HALF\_PI**

**Editor.Math.D2R**

**Editor.Math.R2D**

# Methods

**Editor.Math.deg2rad (degree)**

- `degree` number

Degree to radius.

**Editor.Math.rad2deg (radius)**

- `radius` number

Radius to degree.

**Editor.Math.rad180 (radius)**

- `radius` number

Let radius in -pi to pi.

**Editor.Math.rad360 (radius)**

- `radius` number

Let radius in 0 to 2pi.

**Editor.Math.deg180 (degree)**

- `degree` number

Let degree in -180 to 180.

## **Editor.Math.deg360 (degree)**

- `degree` number

Let degree in 0 to 360.

## **Editor.Math.randomRange (min, max)**

- `min` number
- `max` number

Returns a random floating-point number between min (inclusive) and max (exclusive).

## **Editor.Math.randomRangeInt (min, max)**

- `min` number
- `max` number

Returns a random integer between min (inclusive) and max (exclusive).

## **Editor.Math.clamp (val, min, max)**

- `val` number
- `min` number
- `max` number

Clamps a value between a minimum float and maximum float value.

## **Editor.Math.clamp01 (val)**

- `val` number

Clamps a value between 0 and 1.

## **Editor.Math.calculateMaxRect (out, p0, p1, p2, p3)**

- `out rect`
- `p0 vec2`
- `p1 vec2`
- `p2 vec2`
- `p3 vec2`

## **Editor.Math.lerp (from, to, ratio)**

- `from` number
- `to` number
- `ratio` number

## **Editor.Math.numOfDecimals (val)**

- `val` number

Get number of decimals for decimal part.

## **Editor.Math.numOfDecimalsF (val)**

- `val` number

Get number of decimals for fractional part.

## **Editor.Math.toPrecision (val, precision)**

- `val` number
- `precision` number

## **Editor.Math.bezier (c0, c1, c2, c3, t)**

- `c0` number
- `c1` number
- `c2` number
- `c3` number
- `t` number

## **Editor.Math.solveCubicBezier (c0, c1, c2, c3, x)**

- `c0` number
- `c1` number
- `c2` number
- `c3` number
- `x` number

# **Editor.Selection**

## **Methods**

## **Editor.Selection.register (type)**

- `type` string

## **Editor.Selection.reset ()**

## **Editor.Selection.local ()**

Returns a `Editor.Selection.ConfirmableSelectionHelper` instance.

## **Editor.Selection.confirm ()**

Confirms all current selecting objects, no matter which type they are. This operation may trigger deactivated and activated events.

## **Editor.Selection.cancel ()**

Cancels all current selecting objects, no matter which type they are. This operation may trigger selected and unselected events.

## **Editor.Selection.confirmed (type)**

- `type` string

Check if selection is confirmed.

## **Editor.Selection.select (type, id[, unselectOthers, confirm])**

- `type` string
- `id` string
- `unselectOthers` boolean
- `confirm` boolean

Select item with its id.

## **Editor.Selection.unselect (type, id[, confirm])**

- `type` string
- `id` string
- `confirm` boolean

Unselect item with its id.

## **Editor.Selection.hover (type, id)**

- `type` string
- `id` string

Hover item with its id. If id is null, it means hover out.

## **Editor.Selection.setContext (type, id)**

- `type` string
- `id` string

## **Editor.Selection.patch (type, srcID, destID)**

- `type` string
- `srcID` string
- `destID` string

## **Editor.Selection.clear (type)**

- `type` string

## **Editor.Selection.hovering (type)**

- `type` string

## **Editor.Selection.contexts (type)**

- `type` string

## **Editor.Selection.curActivate (type)**

- `type` string

## **Editor.Selection.curGlobalActivate (type)**

- `type` string

## **Editor.Selection.curSelection (type)**

- `type` string

## **Editor.Selection.filter (items, mode, func)**

- `items` array(string)
- `mode` string - 'top-level', 'deep' and 'name'

- `func` function

# **Editor.Undo**

## **Methods**

**Editor.Undo.undo ()**

**Editor.Undo.redo ()**

**Editor.Undo.add (id, info)**

- `id` string
- `info` object

**Editor.Undo.commit ()**

**Editor.Undo.cancel ()**

**Editor.Undo.collapseTo (index)**

- `index` number

**Editor.Undo.save ()**

**Editor.Undo.clear ()**

**Editor.Undo.reset ()**

**Editor.Undo.dirty ()**

**Editor.Undo.setCurrentDescription (desc)**

- `desc` string

**Editor.Undo.register (id, cmd)**

- `id` string

- `cmd` Editor.Undo.Command

## Class: Editor.Undo.Command

### Instance Methods

**cmd.undo ()**

**cmd.redo ()**

**cmd.dirty ()**

## Editor.Utils

### Methods

**Editor.Utils.padLeft (text, width, ch)**

- `text` string
- `width` number
- `ch` string - The character used to pad

**Editor.Utils.toFixed (value, precision, optionals)**

- `value` number
- `precision` number
- `optionals` number

Implementation of `toFixed()` that treats floats more like decimals

Fixes binary rounding issues (eg. `(0.615).toFixed(2) === '0.61'`) that present problems for accounting- and finance-related software.

**Editor.Utils.formatFrame (frame, frameRate)**

- `frame` number
- `frameRate` number

**Editor.Utils.smoothScale (curScale, delta)**

- `curScale` number
- `delta` number

## **Editor.Utils.wrapError (curScale, delta)**

- `err` Error

## **Editor.Utils.arrayCmpFilter (items, func)**

- `items` array
- `func` function

## **Editor.Utils.fitSize (srcWidth, srcHeight, destWidth, destHeight)**

- `srcWidth` number
- `srcHeight` number
- `destWidth` number
- `destHeight` number

## **Editor.Utils.prettyBytes (num)**

- `num` number

Convert bytes to a human readable string: 1337 → 1.34 kB.

Reference: <https://github.com/sindresorhus/pretty-bytes>

## **Editor.Utils.run (execFile, ...args)**

- `execFile` String
- `...args` ...

run `execFile` with `args`.

# **Editor.i18n**

## **Methods**

### **Editor.i18n.format (text)**

- `text` string

Convert an i18n text `i18n:{id}` to string `{id}`

## **Editor.i18n.formatPath (path)**

- `path` string

Convert an i18n path `i18n:{id1}/i18n:{id2}` to string `{id1}/{id2}`.

## **Editor.i18n.t (key, option)**

- `key` string
- `option` object

Mapping an i18n id to translated text.

## **Editor.i18n.extend (phrases)**

- `phrases` object

Extends the phrases.

## **Editor.i18n.replace (phrases)**

- `phrases` object

Replaces the phrases.

## **Editor.i18n.unset (phrases)**

- `phrases` object

Removes phrases.

## **Editor.i18n.clear ()**

Clear all phrases.

# **Properties**

## **Editor.i18n.polyglot**

Get the polyglot instance.

