**生命周期函数**

脚本组件中有一些特别的函数，不需要手动调用，Cocos Creator会在合适的时间自动调用这些函数。

目前提供给用户的主要回调函数有：

* onLoad

组件最早被调用的回调，一般初始化工作，只执行一次

* Start

在onLoad之后，第一次update之前被调用，一般赋值工作，只执行一次

* Update

组件在激活状态下，每一帧调用一次，在渲染（draw）前调用

* lateUpdate

update之后，在渲染（draw）之后每帧执行

* onDestroy

当组件或者所在节点调用了 destroy()，则会调用 onDestroy 回调。

* onEnable

当组件的 “enabled” 属性从 “false” 变为 “true” 时，或者所在节点的 “active” 属性从 “false” 变为 “true” 并且组件处于“enable”状态时，会激活 onEnable 回调。倘若节点第一次被创建且 enabled 为 true，则会在 onLoad 之后，start 之前被调用。

* onDisable

与“onEnable”相反。当组件的 “enabled” 属性从 “true” 变为 “false” 时，或者所在节点的 “active” 属性从 “true” 变为 “false”并且组件处于“enable”状态时，会激活 onDisable 回调。

**创建、销毁与访问节点**

**创建新节点**

除了在编辑器中可视化创建节点外，Cocos Creator也支持用脚本动态创建节点。可通过new cc.Node()的方式创建节点（Node），之后再给节点添加组件，达到和编辑器可视化添加一致效果。

1. var node = new cc.Node('New Sprite');
2. var sp = node.addComponent(cc.Sprite);
3. sp.spriteFrame = this.sprite;
4. node.parent = this.node;

**复制节点**

如果并不是凭空创建，而是复制场景中已有节点，可以通过cc.instantiate方法完成。参数填入目标节点，返回目标节点的复制节点。

1. var node = cc.instantiate(this.target);
2. node.parent = this.node;
3. node.setPosition(0, 0);

复制出来的节点的所有属性都和原来节点一致

**删除节点**

cc.Node 的实例方法destroy() ，可以销毁指定节点。销毁节点并不会立刻从内存中移除，而是在当前帧逻辑更新结束后，统一执行。当一个节点销毁后，该节点就处于无效状态，防止某些未及时停止的异步回调，可以通过 cc.isValid 判断当前节点是否已经被销毁。

1. this.node.destroy();

**创建预制体**

将要制作成预制体的组件或者组合拖拽至资源管理器中，将自动生成预制体。

预制必须实例化后才能操作里面的节点和组件，预制资源是存放在资源中，并不是节点中。所以在我们使用时，不能通过cc.find()查找，而是通过loadres，或者直接拖拽预制体至变量上

1. var PrefabUrl;
2. cc.loader.loadRes(PrefabUrl,function(errorMessage,loadedResource){});
3. var TipBoxPrefab = cc.instantiate(PrefabUrl);

**访问节点和和其他组件**

获得其他组件：Cocos Creator的整体设计思路是用节点（Node）索引组件脚本，所以我们需要先找到目标组件的对应节点，然后通过节点提供方法“getComponent”获取指定组件。范例代码如下：

1. var label = this.node.getComponent(cc.Label);

第三方或者用户自定义脚本组件则可以通过传入类名（脚本文件名），注意区分大小写。参照如下代码：

1. var label = this.getComponent("HelloScript");

获得所有子节点：节点（cc.Node）的实例属性“children”，可获得指定节点所有子节点的数组

1. var children = this.node.children;
2. for (var i=0;i<children.length;i++){
3. var node = children[i];
4. //...
5. }

获得指定名字子节点：节点（cc.Node）的实例方法“getChildByName”，可获得指定节点指定名字子节点，其中节点名字就是编辑器层级管理器中看到的名字

1. this.node.getChildByName("Child1");

通过名字与层级关系获取子节点：如果没有持有目标节点的父节点，我们可以从场景的根节点开始，通过cc.find方法逐层查找

1. this.bgNode = cc.find("Canvas/Menu/BackGround");

**加载和切换场景**

一般的，Cocos Creator通过cc.director的实例方法loadScene来加载或替换场景

1. cc.director.loadScene("NewScene");

预加载场景：由于加载某些场景需要一些时间，为了避免切换场景时间过长，我们可以提前异步加载场景。这种预加载是异步完成的，不会造成任何卡顿

1. cc.director.preloadScene("new scene", function () {
2. cc.log("scene is preloaded");
3. });

**常用节点和组件接口**

**关闭节点：**

1. this.node.active = false;

**关闭当前节点，和在属性检查器中点掉节点的“√”效果是一致的，这个操作会导致该节点、该节点的所有子节点、该节点上所有的脚本组件、该节点所有子节点上所有的脚本组件失效。**

**激活节点：**

1. this.node.active = true;

**同关闭节点，和在属性检查器中点出节点中“√”效果是一致的。这个操作会导致该节点、该节点所有的子节点但是不包括单独设置过关闭的子节点、该节点上所有的组件但是不包括单独设置过关闭的组件、该节点所有子节点但是不包括单独设置过关闭的子节点上所有的组件但是不包括单独设置过关闭的组件生效。**

**更改节点的父节点：**

1. this.node.parent = parentNode;

**节点的父子关系影响Node中几乎所有的属性，子节点的坐标系会变为父节点的相对坐标系（位置、旋转、缩放），颜色、透明度会叠加到子节点等效果。更改父节点是Cocos Creator常规操作之一。这个操作等价于如下代码：**

1. this.node.removeFromParent(false);
2. parentNode.addChild(this.node);

**更改节点位置：**

**直接为节点的x或者y坐标赋值。**

1. this.node.x = 100;
2. this.node.y = 50;

**调用setPosition：**

**下面两种方式效果是一样的。**

1. this.node.setPosition(100, 50);
2. this.node.setPosition(cc.v2(100, 50));

**更改节点旋转：**

**直接为节点的rotation属性赋值，其中顺时针为正。**

1. this.node.rotation = 90;

**调用setRotation：和上述方式效果一致。**

1. this.node.setRotation(90);

**更改节点缩放：**

**直接修改scaleX或scaleY属性。其中1为100%，负值会导致在该轴渲染颠倒，比如当scaleX为-1时，节点渲染左右颠倒**

1. this.node.scaleX = 2;
2. this.node.scaleY = 2;

**调用setScale：下面两种方式效果是一样的。**

1. this.node.setScale(2);
2. this.node.setScale(2, 2);

**更改颜色：**

**直接修改color属性，color是cc.Color类，Cocos Creator提供了几个基本颜色枚举，比如：**cc.Color.RED

1. this.node.color = cc.Color.RED;

**或者直接设置颜色值**

1. this.node.color = cc.Color（255,255,255,0）;

**设置透明度：**

**直接修改opacity属性，0~255，0为完全透明，255为完全不透明。**

1. this.node.opacity = 128;

**为节点添加组件：**

**addComponent，参数和用法与getComponent一致，为指定节点动态添加指定组件。**

1. this.addComponent(cc.Sprite);