[Cocos Creator 模块化脚本](https://www.1024sou.com/article/292987.html)

收录于 2021-10-06 04:25:07

查看 1756 次

[Cocos Creator](https://www.1024sou.com/search?q=Cocos%20Creator)

Cocos Creator 允许你将代码拆分成多个脚本文件，并且让它们相互调用。这个步骤简称为 **模块化**。

模块化使你可以在 Cocos Creator 中引用其它脚本文件：

* 访问其它文件导出的参数
* 调用其它文件导出的方法
* 使用其它文件导出的类型
* 使用或继承其它 Component

Cocos Creator 中的 JavaScript 使用和 Node.js 几乎相同的 CommonJS 标准来实现模块化，简单来说：

* 每一个单独的脚本文件就构成一个模块
* 每个模块都是一个单独的作用域
* 以 **同步** 的 require 方法来引用其它模块
* 设置 module.exports 为导出的变量

**当你在脚本中声明了一个组件，Creator 会默认把它导出，其它脚本直接 require 这个模块就能使用这个组件。**

// Rotate.js

cc.Class({

extends: cc.Component,

// ...

});

// SinRotate.js

var Rotate = require("Rotate");

var SinRotate = cc.Class({

extends: Rotate,

update: function (dt) {

this.rotation += this.speed \* Math.sin(dt);

}

});

模块里不单单能定义组件，实际上你可以导出任意 JavaScript 对象。假设有个脚本 config.js：

// config.js - v2

var cfg = {

moveSpeed: 10,

version: "0.15",

showTutorial: true,

load: function () {

// ...

}

};

cfg.load();

module.exports = cfg;

现在如果我们要在其它脚本中访问 cfg 对象：

// player.js

var config = require("config");

cc.log("speed is", config.moveSpeed);

**module.exports 的默认值：**  
当你的 module.exports 没有任何定义时，Creator 会自动优先将 exports 设置为脚本中定义的 Component。如果脚本没定义 Component 但是定义了别的类型的 CCClass，则自动把 exports 设为定义的 CCClass。

导出变量

module.exports 默认是一个空对象（{}），可以直接往里面增加新的字段。

// foobar.js:

module.exports.foo = function () {

cc.log("foo");

};

module.exports.bar = function () {

cc.log("bar");

};

// test.js:

var foobar = require("foobar");

foobar.foo(); // "foo"

foobar.bar(); // "bar"

module.exports 的值可以是任意 JavaScript 类型。

// foobar.js:

module.exports = {

FOO: function () {

this.type = "foo";

},

bar: "bar"

};

// test.js:

var foobar = require("foobar");

var foo = new foobar.FOO();

cc.log(foo.type); // "foo"

cc.log(foobar.bar); // "bar"