When you invoke the constructor function with new, the following happens inside the function:

• An empty object is created and referenced by **this** variable, inheriting the prototype of the function.

• Properties and methods are added to the object referenced by this.

• The newly created object referenced by this is returned at the end implicitly (if no other object was returned explicitly).

[复制代码](javascript:void(0);)

var Person = function (name) {

this.name = name;

this.say = function () {

return "I am " + this.name;

};

};

var adam = new Person("Adam");

adam.say(); // "I am Adam"

[复制代码](javascript:void(0);)

**Note**

reusable members, such as methods, should go to the prototype.

Person.prototype.say = function () {

return "I am " + this.name;

};

**Constructor's Return Values**

When invoked with new, a constructor function always returns an object inheriting from the constructor's prototype.

[复制代码](javascript:void(0);)

var Objectmaker = function () {

// this `name` property will be ignored

// because the constructor

// decides to return another object instead

this.name = "This is it";

// creating and returning a new object

var that = {};

that.name = "And that's that";

return that;

};

// test

var o = new Objectmaker();

console.log(o.name); // "And that's that"

[复制代码](javascript:void(0);)

You have the freedom to return any object in your constructors, as long as it's an object. Attempting to return something that's not an object (like a string or a boolean false, for example) will not cause an error but will simply be ignored, and the object referenced by this will be returned instead.