**Basic concept**

Values can be

properties: primitives or other objects

methods: functions

User-defined native objects are mutable at any time.

Object literal notation is ideal for this type of on-demand object creation.

Even the simplest {} object already has properties and methods inherited from Object.prototype.

[复制代码](javascript:void(0);)

var dog = {

name: "Benji",

getName: function () {

return this.name;

}

};

[复制代码](javascript:void(0);)

1. The Object Literal Syntax

• Wrap the object in curly braces ({ and }).

• Comma-delimit the properties and methods inside the object. A trailing comma after the last name-value pair is allowed but produces errors in IE, so don't use it.

• Separate property names and property values with a colon.

• When you assign the object to a variable, don't forget the semicolon after the closing }.

1. Objects from a Constructor

[复制代码](javascript:void(0);)

// one way -- using a literal

var car = {goes: "far"};

// another way -- using a built-in constructor

// warning: this is an antipattern

var car = new Object();

car.goes = "far";

[复制代码](javascript:void(0);)

1. Object Constructor Catch

Don't use new Object(); use the simpler and reliable object literal instead.

[复制代码](javascript:void(0);)

// Warning: antipatterns ahead

// an empty object

var o = new Object();

console.log(o.constructor === Object); // true

// a number object

var o = new Object(1);

console.log(o.constructor === Number); // true

console.log(o.toFixed(2)); // "1.00"

// a string object

var o = new Object("I am a string");

console.log(o.constructor === String); // true

// normal objects don't have a substring()

// method but string objects do

console.log(typeof o.substring); // "function"

// a boolean object

var o = new Object(true);

console.log(o.constructor === Boolean); // true

[复制代码](javascript:void(0);)