# Object 类型

到目前为止，我们看到的大多数引用类型值都是 Object 类型的实例；而且， Object 也是ECMAScript 中使用最多的一个类型。虽然 Object 的实例不具备多少功能，但对于在应用程序中存储和传输数据而言，它们确实是非常理想的选择。

## 创建方式

创建 Object 实例的方式有两种。第一种是使用 **new 操作符**后跟 Object 构造函数，如下所示：

**var person = new Object();**

**person.name = "Nicholas";**

**person.age = 29;**

另一种方式是使用**对象字面量表示法**。对象字面量是对象定义的一种简写形式，目的在于简化创建包含大量属性的对象的过程。下面这个例子就使用了对象字面量语法定义了与前面那个例子中相同的person 对象：

**var person = {**

**name : "Nicholas",**

**age : 29**

**};**

我们定义了 name 属性，之后是一个冒号，再后面是这个属性的值。在对象字面量中，使用逗号来分隔不同的属性，因此 "Nicholas" 后面是一个逗号。但是，在 age 属性的值 29 的后面**不能添加逗号**，因为 age 是这个对象的最后一个属性。在最后一个属性后面添加逗号，会在 IE7 及更早版本和Opera 中导致错误。

在使用对象字面量语法时，属性名也可以使用字符串，如下面这个例子所示。

**var person = {**

**"name" : "Nicholas",**

**"age" : 29,**

**};**

这个例子会创建一个对象，包含三个属性： name 、 age 。另外，使用对象字面量语法时，如果留空其花括号，则可以定义只包含默认属性和方法的对象，如下所示：

**var person = {}; //与 与 new Object() 相同**

**person.name = "Nicholas";**

**person.age = 29;**

**在通过对象字面量定义对象时，实际上不会调用 Object 构造函数**。

虽然可以使用前面介绍的任何一种方法来定义对象，但开发人员更青睐对象字面量语法，因为这种语法要求的代码量少，而且能够给人封装数据的感觉。实际上，对象字面量也是向函数传递大量可选参数的首选方式，例如：

**function displayInfo(args) {**

**var output = "";**

**if (typeof args.name == "string"){**

**output += "Name: " + args.name + "\n";**

**}**

**if (typeof args.age == "number") {**

**output += "Age: " + args.age + "\n";**

**}**

**alert(output);**

**}**

**displayInfo({**

**name: "Nicholas",**

**age: 29**

**});**

**displayInfo({**

**name: "Greg"**

**});**

在这个例子中，函数 displayInfo() 接受一个名为 args 的参数。这个参数可能带有一个名为 name或 age 的属性，也可能这两个属性都有或者都没有。在这个函数内部，我们通过 typeof 操作符来检测每个属性是否存在，然后再基于相应的属性来构建一条要显示的消息。然后，我们调用了两次这个函数，每次都使用一个对象字面量来指定不同的数据。这两次调用传递的参数虽然不同，但函数都能正常执行。

## 属性的访问

一般来说，访问对象属性时使用的都是**点表示法**，这也是很多面向对象语言中通用的语法。不过，在 JavaScript **也可以使用方括号表示法来访问对象的属性**。在使用方括号语法时，应该将要访问的属性以字符串的形式放在方括号中，如下面的例子所示。

**alert(person["name"]); //"Nicholas"**

**alert(person.name); //"Nicholas"**

从功能上看，这两种访问对象属性的方法没有任何区别。但**方括号语法的主要优点是可以通过变量来访问属性**，例如：

**var propertyName = "name";**

**alert(person[propertyName]); //"Nicholas"**

如果属性名中包含会导致语法错误的字符，或者属性名使用的是关键字或保留字，也可以使用方括号表示法。例如：

**person["first name"] = "Nicholas";**

由于 "first name" 中包含一个空格，所以不能使用点表示法来访问它。然而，属性名中是可以包含非字母非数字的，这时候就可以使用方括号表示法来访问它们。

通常，除非必须使用变量来访问属性，否则我们建议使用点表示法。

## 对象的遍历

**for...in 语句**以任意顺序遍历一个对象的可枚举属性。对于每个不同的属性，语句都会被执行。每次迭代时，分配的是属性名

## 理解this

**var obj = {**

**foo: function () {}**

**};**

**var foo = obj.foo;**

// 写法一

**obj.foo()**

// 写法二

**foo()**

上面代码中，虽然obj.foo和foo指向同一个函数，但是执行结果可能不一样。请看下面的例子。

**var obj = {**

**foo: function () { console.log(this.bar) },**

**bar: 1**

**};**

**var foo = obj.foo;**

**var bar = 2;**

**obj.foo() // 1**

**foo() // 2**

这种差异的原因，就在于函数体内部使用了this关键字。this指的是函数运行时所在的环境。对于obj.foo()来说，foo运行在obj环境，所以this指向obj；对于foo()来说，foo运行在全局环境，所以this指向全局环境。在全局作用域下，this指向全局对象。在浏览器中全局对象就是window对象。所以，两者的运行结果不一样。