17.9.2 使用 toString 检测数据类型

使用 toString()方法可以设计一种更安全的检测 JavaScript 数据类型的方法，用户还可以根据开发需要进一步补充检测类型的范围。

设计思路：

首先，仔细分析不同类型对象的 toString()方法返回值，会发现由 Object 对象定义的 toString()方法返回的字符串形式总是：

```
[object class]
```

其中 object 表示对象的通用类型，class 表示对象的内部类型，内部类型的名称与该对象的构造函数名对应。例如，Array 对象的 class 为“Array”，Function 对象的 class 为“Function”，Date 对象的 class 为“Date”，内部 Math 对象的 class 为“Math”，所有 Error 对象（包括各种 Error 子类的实例）的 class 为“Error”。

客户端 JavaScript 的对象和由 JavaScript 实现定义的其他所有对象都具有预定义的特定 class 值，如“Window”、“Document”和“Form”等。用户自定义对象的 class 为“Object”。

class 值提供的信息与对象的 constructor 属性值相似，但是 class 值是以字符串的形式提供这些信息的，这在特定的环境中是非常有用的。如果使用 typeof 运算符来检测，则所有对象的 class 值都为“Object”或“Function”。所以不能够提供有效信息。

但是，要获取对象的 class 值的唯一方法是必须调用 Object 的原型方法 toString()，因为很多类型对象都会重置 Object 的 toString()方法，所以不能直接调用对象的 toString()方法。

例如，下面对象的 toString()方法返回的就是当前 UTC 时间字符串，而不是字符串“[object Date]”。

```
var d = new Date();
alert(d.toString());              // 返回当前 UTC 时间字符串
```

调用 Object 的 toString()原型方法，可以通过调用 Object.prototype.toString 对象的默认 toString()函数，再调用该函数的 apply()方法在想要检测的对象上执行即可。例如，结合上面的对象 d，具体实现代码如下：

```
var d = new Date();
var m = Object.prototype.toString;
alert(m.apply(d));                // 返回字符串" [object Date] "
```

明白了上面的技术细节，下面就是一个比较完整的数据类型安全检测方法源代码：

```
// 安全检测 JavaScript 基本数据类型和内置对象
// 参数：o 表示检测的值
// 返回值：返回字符串"undefined"、"number"、"boolean"、"string"、"function"、
    "regexp"、"array"、"date"、"error"、"object"或"null"
function typeOf(o){
    var _toString = Object.prototype.toString;
    // 获取对象的 toString()方法引用
    // 列举基本数据类型和内置对象类型，你还可以进一步补充该数组的检测数据类型范围
    var _type ={
        "undefined" : "undefined",
        "number" : "number",
        "boolean" : "boolean",
        "string" : "string",
        "[object Function]" : "function",
        "[object RegExp]" : "regexp",
        "[object Array]" : "array",
        "[object Date]" : "date",
        "[object Error]" : "error"
    }
    return _type[typeof o] || _type[_toString.call(o)] || (o ? "object" :
            "null");
    // 通过把值转换为字符串，然后匹配返回字符串中是否包含特定字符进行检测
}
```

应用示例：

```
var a = Math.abs;
alert(typeOf(a));                    // 返回字符串"function"
```

上述方法适用于 JavaScript 基本数据类型和内置对象，但是对于自定义对象是无效的。因为自定义对象被转换为字符串后，返回的值是没有规律的，且不同浏览器返回值也是不同的。因此，如果要检测非内置对象，只能够使用 constructor 属性和 instaceof 运算符来实现。