标准对象

在JavaScript的世界里，一切都是对象。

但是某些对象还是和其他对象不太一样。为了区分对象的类型，我们用`typeof`操作符获取对象的类型，它总是返回一个字符串：

```
typeof 123; // 'number'
typeof NaN; // 'number'
typeof 'str'; // 'string'
typeof true; // 'boolean'
typeof undefined; // 'undefined'
typeof Math.abs; // 'function'
typeof null; // 'object'
typeof []; // 'object'
typeof {}; // 'object'
```

可见，`number`、`string`、`boolean`、`function`和`undefined`有别于其他类型。特别注意`null`的类型是`object`，`Array`的类型也是`object`，如果我们用`typeof`将无法区分出`null`、`Array`和通常意义上的object——`{}`。

包装对象

除了这些类型外，JavaScript还提供了包装对象，熟悉Java的小伙伴肯定很清楚`int`和`Integer`这种暧昧关系。

`number`、`boolean`和`string`都有包装对象。没错，在JavaScript中，字符串也区分`string`类型和它的包装类型。包装对象用`new`创建：

```
var n = new Number(123); // 123,生成了新的包装类型
var b = new Boolean(true); // true,生成了新的包装类型
var s = new String('str'); // 'str',生成了新的包装类型
```

虽然包装对象看上去和原来的值一模一样，显示出来也是一模一样，但他们的类型已经变为`object`了！所以，包装对象和原始值用`===`比较会返回`false`：

```
typeof new Number(123); // 'object'
new Number(123) === 123; // false

typeof new Boolean(true); // 'object'
new Boolean(true) === true; // false

typeof new String('str'); // 'object'
new String('str') === 'str'; // false
```

所以*闲的蛋疼也不要使用包装对象*！尤其是针对`string`类型！！！

如果我们在使用`Number`、`Boolean`和`String`时，没有写`new`会发生什么情况？

此时，`Number()`、`Boolean`和`String()`被当做普通函数，把任何类型的数据转换为`number`、`boolean`和`string`类型（注意不是其包装类型）：

```
var n = Number('123'); // 123，相当于parseInt()或parseFloat()
typeof n; // 'number'

var b = Boolean('true'); // true
typeof b; // 'boolean'

var b2 = Boolean('false'); // true! 'false'字符串转换结果为true！因为它是非空字符串！
var b3 = Boolean(''); // false

var s = String(123.45); // '123.45'
typeof s; // 'string'
```

是不是感觉头大了？这就是JavaScript特有的催眠魅力！

总结一下，有这么几条规则需要遵守：

- 不要使用`new Number()`、`new Boolean()`、`new String()`创建包装对象；
- 用`parseInt()`或`parseFloat()`来转换任意类型到`number`；
- 用`String()`来转换任意类型到`string`，或者直接调用某个对象的`toString()`方法；
- 通常不必把任意类型转换为`boolean`再判断，因为可以直接写`if (myVar) {...}`；
- `typeof`操作符可以判断出`number`、`boolean`、`string`、`function`和`undefined`；
- 判断`Array`要使用`Array.isArray(arr)`；
- 判断`null`请使用`myVar === null`；
- 判断某个全局变量是否存在用`typeof window.myVar === 'undefined'`；
- 函数内部判断某个变量是否存在用`typeof myVar === 'undefined'`。

最后有细心的同学指出，任何对象都有`toString()`方法吗？`null`和`undefined`就没有！确实如此，这两个特殊值要除外，虽然`null`还伪装成了`object`类型。

更细心的同学指出，`number`对象调用`toString()`报SyntaxError：

```
123.toString(); // SyntaxError
```

遇到这种情况，要特殊处理一下：

```
123..toString(); // '123', 注意是两个点！
(123).toString(); // '123'
```