Number类型

Number 是与数字值对应的引用类型。要创建 Number 对象，可以在调用 Number 构造函数时向其中传递相应的数值。下面是一个例子。

```
var numberObject = new Number(10);
```

除了继承的方法之外，Number 类型还提供了一些用于将数值格式化为字符串的方法。其中，toFixed()方法会按照指定的小数位返回数值的字符串表示，例如：

```
var num = 10;
alert(num.toFixed(2));      //"10.00"
```

toFixed()方法可以表示带有 0 到 20 个小数位的数值。但这只是标准实现的范围，有些浏览器也可能支持更多位数。

另外可用于格式化数值的方法是 toExponential()，该方法返回以指数表示法（也称 e 表示法）表示的数值的字符串形式。与 toFixed()一样，toExponential()也接收一个参数，而且该参数同样也是指定输出结果中的小数位数。看下面的例子。

```
var num = 10;
alert(num.toExponential(1));   //"1.0e+1"
```

以上代码输出了"1.0e+1"；不过，这么小的数值一般不必使用 e 表示法。如果你想得到表示某个数值的最合适的格式，就应该使用 toPrecision()方法。

```
var num = 99;
alert(num.toPrecision(1));      //"1e+2"
alert(num.toPrecision(2));      //"99"
alert(num.toPrecision(3));      //"99.0"
```

toPrecision()方法可以表现 1 到 21 位小数。某些浏览器支持的范围更大，但这是典型实现的范围。

与 Boolean 对象类似， Number 对象也以后台方式为数值提供了重要的功能。但与此同时，我们仍然不建议直接实例化 Number 类型，而原因与显式创建

Boolean 对象一样。具体来讲，就是在使用typeof 和 instanceof 操作符测试基本类型数值与引用类型数值时，得到的结果完全不同，如下面的例子所示。

```
var numberObject = new Number(10);
var numberValue = 10;
alert(typeof numberObject);   //"object"
alert(typeof numberValue);    //"number"
alert(numberObject instanceof Number);  //true
alert(numberValue instanceof Number);   //false
```