[**JS 数据类型转换**](http://www.cnblogs.com/446557021/archive/2011/10/13/2211043.html)

JS 数据类型转换 方法主要有三种

转换函数、强制类型转换、利用js变量弱类型转换。

1. 转换函数：

js提供了parseInt()和parseFloat()两个转换函数。前者把值转换成整数，后者把值转换成浮点数。只有对String类型调用这些方法，这两个函数才能正确运行；对其他类型返回的都是NaN(Not a Number)。

在判断字符串是否是数字值前，parseInt()和parseFloat()都会仔细分析该字符串。parseInt()方法首先查看位置0处的 字符，判断它是否是个有效数字；如果不是，该方法将返回NaN，不再继续执行其他操作。但如果该字符是有效数字，该方法将查看位置1处的字符，进行同样的 测试。这一过程将持续到发现非有效数字的字符为止，此时parseInt()将把该字符之前的字符串转换成数字。

例如，如果要把字符串 "1234blue "转换成整数，那么parseInt()将返回1234，因为当它检测到字符b时，就会停止检测过程。字符串中包含的数字字面量会被正确转换为数字，因此 字符串 "0xA "会被正确转换为数字10。不过，字符串 "22.5 "将被转换成22，因为对于整数来说，小数点是无效字符。一些示例如下：

parseInt("1234blue");   //returns   1234   
parseInt("0xA");   //returns   10   
parseInt("22.5");   //returns   22   
parseInt("blue");   //returns   NaN

parseInt()方法还有基模式，可以把二进制、八进制、十六进制或其他任何进制的字符串转换成整数。基是由parseInt()方法的第二个参数指定的，所以要解析十六进制的值，需如下调用parseInt()方法：   
parseInt("AF",   16);   //returns   175   
当然，对二进制、八进制，甚至十进制（默认模式），都可以这样调用parseInt()方法：   
parseInt("10",   2);   //returns   2   
parseInt("10",   8);   //returns   8   
parseInt("10",   10);   //returns   10   
如果十进制数包含前导0，那么最好采用基数10，这样才不会意外地得到八进制的值。例如：   
parseInt("010");   //returns   8   
parseInt("010",   8);   //returns   8   
parseInt("010",   10);   //returns   10   
在这段代码中，两行代码都把字符串 "010 "解析成了一个数字。第一行代码把这个字符串看作八进制的值，解析它的方式与第二行代码（声明基数为8）相同。最后一行代码声明基数为10，所以iNum3最后等于10。

parseFloat()方法与parseInt()方法的处理方式相似，从位置0开始查看每个字符，直到找到第一个非有效的字符为止，然后把该字 符之前的字符串转换成数字。不过，对于这个方法来说，第一个出现的小数点是有效字符。如果有两个小数点，第二个小数点将被看作无效的， parseFloat()方法会把这个小数点之前的字符串转换成数字。这意味着字符串 "22.34.5 "将被解析成22.34。   
使用parseFloat()方法的另一不同之处在于，字符串必须以十进制形式表示浮点数，而不能用八进制形式或十六进制形式。该  
方法会忽略前导0，所以八进制数0908将被解析为908。对于十六进制数0xA，该方法将返回NaN，因为在浮点数中，x不是有效字符。此外，parseFloat()也没有基模式。

下面是使用parseFloat()方法的示例：   
parseFloat("1234blue");   //returns   1234.0   
parseFloat("0xA");   //returns   NaN   
parseFloat("22.5");   //returns   22.5   
parseFloat("22.34.5");   //returns   22.34   
parseFloat("0908");   //returns   908   
parseFloat("blue");   //returns   NaN

2. 强制类型转换

还可使用强制类型转换（type casting）处理转换值的类型。使用强制类型转换可以访问特定的值，即使它是另一种类型的。  
ECMAScript中可用的3种强制类型转换如下：   
Boolean(value)——把给定的值转换成Boolean型；   
Number(value)——把给定的值转换成数字（可以是整数或浮点数）；   
String(value)——把给定的值转换成字符串。   
用这三个函数之一转换值，将创建一个新值，存放由原始值直接转换成的值。这会造成意想不到的后果。   
当要转换的值是至少有一个字符的字符串、非0数字或对象（下一节将讨论这一点）时，Boolean()函数将返回true。如果该值是空字符串、数字0、undefined或null，它将返回false。

可以用下面的代码段测试Boolean型的强制类型转换。

Boolean("");   //false   –   empty   string   
Boolean("hi");   //true   –   non-empty   string   
Boolean(100);   //true   –   non-zero   number   
Boolean(null);   //false   -   null   
Boolean(0);   //false   -   zero   
Boolean(new   Object());   //true   –   object

Number()的强制类型转换与parseInt()和parseFloat()方法的处理方式相似，只是它转换的是整个值，而不是部分值。还记 得吗，parseInt()和parseFloat()方法只转换第一个无效字符之前的字符串，因此 "4.5.6 "将被转换为 "4.5 "。用Number()进行强制类型转换， "4.5.6 "将返回NaN，因为整个字符串值不能转换成数字。如果字符串值能被完整地转换，Number()将判断是调用parseInt()方法还是调用 parseFloat()方法。下表说明了对不同的值调用Number()方法会发生的情况：

用　　法 结　　果   
Number(false)   0   
Number(true)   1   
Number(undefined)   NaN   
Number(null)   0   
Number( "5.5 ")   5.5   
Number( "56 ")   56   
Number( "5.6.7 ")   NaN   
Number(new   Object())   NaN   
Number(100)   100

最后一种强制类型转换方法String()是最简单的，因为它可把任何值转换成字符串。要执行这种强制类型转换，只需要调用作为参数传递进来的值的 toString()方法，即把1转换成   "1 "，把true转换成 "true "，把false转换成 "false "，依此类推。强制转换成字符串和调用toString()方法的唯一不同之处在于，对null或undefined值强制类型转换可以生成字符串而不引 发错误：

var   s1   =   String(null);   //"null"   
var   oNull   =   null;   
var   s2   =   oNull.toString();   //won’t   work,   causes   an   error

3. 利用js变量弱类型转换

举个小例子，一看，就会明白了。  
<script>   
var   str= '012.345 ';   
var   x   =   str-0;   
x   =   x\*1;  
</script>