aardio 范例: 怪异编码转换

```
//怪异编码转换
/*
这种 ANSI 怪异模式已经被现代软件淘汰，
如果没有特别的原因，不必要浪费时间去理解这种怪异的工作模式，
有这个时间，去学习用机器码写软件可能更有意义。
*/
import console;
import com.interface.IMultiLanguage2;
var mlang = com.interface.IMultiLanguage2.Create()

//怪异模式转换，一句代码、指定一个编码参数就能自动修复乱码
var str = mlang.fromto("锘挎",936);
console.log(str);

//这个函数的用法很任性，这样也能修复这个字符串的乱码
var str = mlang.fromto("锘挎",,936);
console.log(str);

//怪异模式转换，修复乱码（ 950 编码，被错误地当作936编码，然后再一次错误地转换为了 65001 编码）
var str = mlang.fromto("虏鈥锛传箋",936,-950);
/*
上面参数中的负数-950表示阻止转换为936编码的字符串转换为950编码，
并且将已经转换为936编码的字符串认作950编码进行下一轮转换，
下一轮转换未指定目标参数，则默认转为 UTF-8编码（代码页：65001）
*/
console.log(str)

/*
如果编码转换后显示为了完全不相关的字符（不是丢失不能显示的字符），这就说明发生了错误的转换，
ANSI 编码的软件会导致这种古怪的错误，只能用怪异模式转换来还原乱码。

aardio 拥有强大的 UTF 标记功能，能有效地避免怪异模式，在变更存储编码时保持显示字符不变。
aardio 中的 string.fromto() 等默认编码转换函数都会自动阻止怪异模式转换。

但如果你是试图还原来自外部程序怪异模式制造的乱码，
aardio 在com.interface.IMultiLanguage2 中提供了允许怪异模式转换的 fromto() 函数。
这个函数会在转换前清除 UTF 标记，允许将一个存储编码"错"认为另外一个存储编码，
并且在转换过程中，会尝试修复 UTF-8 编码。
*/

console.pause();
```

Markdown 格式