aardio 范例: 基数排序

```
import console;

/*
基数排序从低位到高位进行,使得最后一次计数排序完成后,数组有序。
例如比较时间,先按日排序,再按月排序,最后按年排序,仅需排序三次。
但是如果先排序高位就没这么简单了。

基数排序源于老式穿孔机,排序器每次只能看到一个列。
其原理在于对于待排序的数据,整体权重未知的情况下,
先按权重小的因子排序,然后按权重大的因子排序。

基数排序更适合用于对时间、字符串等这些整体权值未知的数据进行排序。
又或者所有的数值都是以字符串形式存储,就象穿孔机一样,每次只能对一列进行排序。
这时候基数排序也适用,例如:对{"193";"229";"233";"215"}进行排序。
*/

//计数排序算法
var radixSort = function( array ,maxlen){

    //aardio 在字符串索引越界时,会返回0,这使基数排序的实现更加简单。
    //我们首先找出最大的排序长度,然后对于不足此长度的字符串,尾部都假定以 0 补齐。
    //对于超出此长度的位在比较时忽略
    if(!maxlen){
        maxlen =0;
        for(i=1;#array;1){
            maxlen = math.max(maxlen,#array[i] )
        }
    }
    //else{
        //最大排序长度也可以从参数中传过来,这样就不用遍历所有字符串了
    //}

    //从字符串的最后一位开始,到第一位
    for(pos=maxlen;1;-1){
        //按当前位的字节码计数排序
        var arraySorted ={};

        var count = {};
        for(i=0;256 ){
            count[i] = 0;
        }

        var bytecode;
        for(i=1;#array;1){
            //如果 pos 大于字符串长度,aardio 会返回 0,这使基数排序的实现更容易
            bytecode = array[i][pos] ;
            count[ bytecode ] ++; //count[n] 包含等于n的个数
        }

        //统计位置
        for(i=1;256;1){
            count[i] += count[i-1]; //count[i] 包含小于等于i的个数
        }

        var n;
        for(i=#array;1;-1){
            n = array[i][pos]
            arraySorted[ count[n] ] = array[i];
            count[n]--;//防止相同的元素n再次出现,将计数减一
        }

        array = arraySorted;
    }

    return array;
}

console.log("----------------");
console.log("基数排序( 线性时间排序 )");
console.log("----------------");

var array ={"Python";"aardio";"193";"229";"233";"215";"Hello Word";"abc";"abcd";"xd";"adcd";"eddd";"ah";"ai";"aj";"ajkk"};

//排序
array = radixSort(array  );

//输出结果
for(i=1;#array;1){
    console.log( array[i] );
}

execute("pause");
```