aardio 范例: 编译运行 C# 代码，.NET 类型检测

```
//aardio 编译运行 C# 代码，.NET 类型检测
/*点开这里查看编译程序集代码{{*/
import dotNet;
import console.int;
console.open();

//创建  C# 语言编译器
var compiler = dotNet.createCompiler("C#");

//引入依赖 DLL 程序集(不要内存加载)，System.dll 默认已引入
compiler.Reference("System.dll");

//可选指定编译参数 https://learn.microsoft.com/dotnet/csharp/language-reference/compiler-options
//compiler.Parameters.CompilerOptions = "/optimize /unsafe" ;

//设置 C# 源码（ 注释赋值为字符串，首尾星号数目要匹配 ）
compiler.Source = /******
<? /* Source 值为字符串则启用模板语法 https://www.aardio.com/zh-cn/doc/language-reference/templating/syntax.html */ ?>
using System;

namespace CSharpLibrary
{
    public class Util
    {
        public static object TestDataType<T>(T parameter)
        {
            //使用 typeof 获取参数的类型
            Type parameterType = typeof(T);

            //内置类型别名表:https://learn.microsoft.com/zh-cn/dotnet/csharp/language-reference/builtin-types/built-in-types
            Console.WriteLine("aardio 参数对应 C# 类型: \t{0} \n", parameterType);

            byte [] buffer = new byte[]{0x60,0x61};
            return buffer;
        }
    }
}
******/

//编译并返回程序集，可选指定输出本地 DLL文件。
var assembly = compiler.CompileOrFail(/*"/output.dll"*/);
//Win10 已自带 .NET 4.x （ CLR 4.0 ） ，CLR 2.0 编译器不支持 var ，lambda 这些语法。

//自内存程序集导入名字空间
assembly.import("CSharpLibrary");
/*}}*/

//要测试 C# 类型的 aardio 对象
var value = raw.buffer("测试字节数组");

//使用 C# 编写的类构造对象实例
var netUtil = CSharpLibrary.Util();

//调用实时编译的 C# 函数（模板参数，可传任意类型）
var result = netUtil.TestDataType(value);

console.log("C# 返回值的 aardio 类型: ",type(result),'\n\n\n' );

/*
aardio / C# 常用类型对应关系
--------------------------------------------------
aardio | C#
bool   | bool
string | string
buffer | byte[]
number(整数) | int
number(小数) | double
number数组   | double[]
string 数组  | string[] 底层是 COM 中的 BSTR 数组
其他数组     | object[] 底层是 COM 中的 Variant 变体类型数组
table        | object,System.__ComObject,dynamic
pointer      | 不支持(只能先转为数值)

.NET 特殊类型处理
--------------------------------------------------

1、enum 枚举:
在 aardio .NET 的枚举自动转换为数值,
aardio 数值也可自动转换为 .NET 枚举参数。

2、颜色数值
.NET 的 System.Drawing.Color 在 aardio 则会自动转换为 ARGB 格式的颜色数值。
调用 .NET 时 ARGB 格式的颜色数值也能自动转换为 System.Drawing.Color 对象。
```

注意 GDI+ 使用 ARGB 格式颜色值，与 gdip库，plus 控件等兼容。

3、aardio 函数
aardio 函数可自动转换为 .NET 委托、事件所需要的委托类型。
并且自动处理调用参数与返回值的数据类型。

但在 .NET 中通过 InvokeMember,dynamic 直接调用 aardio 对象成员时，
回调参数中的 .NET 原生对象为原生 COM 对象，需要自己调用 dotNet.object 转换为更易操作的 dotNet.object 对象。

4、指针句柄
.NET 中的 System.IntPtr,System.UIntPtr 类型在 aardio 中会自动转换为整数值，
aardio 中的指针类型（pointer）必须使用 tonumber() 函数转换为数值才能传入 .Net。
HWND 在 aardio 以整数值表示，可以直接传入 .Net。

dotNet.object 对象封包、解包原理

所有原生 .NET 中的值在 aardio 中分为两类：
null值、数值、字符串、枚举、 System.Drawing.Color 等简单值类型，以及这些值类型的数组可以直接交换。
其他原生 .NET 对象在 aardio 中存为 com.NETObject 对象（对应 .NET 中的 System.__ComObject 类型）。

com.NetObject 分为：
1、普通 .NET 对象，传入 com.IsNetObject() 返回1
2、封包其他原始 .NET 对象的 DispatchableObject 对象，传入 com.IsNetObject() 返回2
一些 aardio 无法直接转换的 .NET 对象（例如 struct,ValueTuple,交错数组）会被自动封包到 DispatchableObject 内 (在传回 .NET 时会自动解包)。

.NET / aardio 底层交互基于 COM 接口，
但 COM 接口 难以兼容 .NET 对象复杂的数据类型与语法特性（例如无法支持函数重载），
aardio 为了解决这个问题，将 com.NetObject 自动封包为了更易使用的 dotNet.object。

普通 .NET 对象自动封装为 dotNet.object 以后就可以直接使用。

如果 DispatchableObject 存储的是Primitive,enum,string 类型或这些类型的普通数组。
则在自动封装为 dotNet.object 以后，可以使用 Value 属性读写 .NET 对象原始值。

在 aardio 中可调用以下函数创建指定 .NET 类型的 dotNet.object 对象：

dotNet.object(value,byRef)
将参数 @value 指定值或数组转换为 .NET 对象。@byRef 参数值为 true 则支持.Net 输出或引用参数。

dotNet.buffer(size,value) 或 dotNet.buffer(value)
等价于调用 dotNet.object(raw.buffer(...),true) 。
返回封包 buffer 的 dotNet.object，在 .NET 中可作为 byte[] 使用，支持.Net 输出或引用参数。

dotNet.byte(value,byRef)
将参数 @value 指定的数值或数组转换为 8 位整型数值。@byRef 参数值为 true 则支持.Net 输出或引用参数。

dotNet.ubyte(value,byRef)
将参数 @value 指定的数值或数组转换为 8 位无符号整型数值。@byRef 参数值为 true 则支持.Net 输出或引用参数。

dotNet.word(value,byRef)
将参数 @value 指定的数值或数组转换为 16 位整型数值。@byRef 参数值为 true 则支持.Net 输出或引用参数。

dotNet.uword(value,byRef)
将参数 @value 指定的数值或数组转换为 16 位无符号整型数值。@byRef 参数值为 true 则支持.Net 输出或引用参数。

dotNet.int(value,byRef)
将参数 @value 指定的数值或数组转换为 32 位整型数值。@byRef 参数值为 true 则支持.Net 输出或引用参数。

dotNet.uint(value,byRef)
将参数 @value 指定的数值或数组转换为 32 位无符号整型数值。@byRef 参数值为 true 则支持.Net 输出或引用参数。

dotNet.long(value,byRef)
将参数 @value 指定的数值或数组转换为 64 位整型数值。@byRef 参数值为 true 则支持.Net 输出或引用参数。

dotNet.ulong(value,byRef)
将参数 @value 指定的数值或数组转换为 64 位无符号整型数值。@byRef 参数值为 true 则支持.Net 输出或引用参数。

dotNet.float(value,byRef)
将参数 @value 指定的数值或数组转换为 32 位浮点数值。@byRef 参数值为 true 则支持.Net 输出或引用参数。

dotNet.double(value,byRef)
将参数 @value 指定的数值或数组转换为 64 位浮点数值。@byRef 参数值为 true 则支持.Net 输出或引用参数。

以上函数会将所有对应的参数值存为 .NET 对象 DispatchableObject 以后，
再封包为 dotNet.object 对象。

即使简单的值类型也会转换为 dotNet.object 对象，
这可以让 aardio 直接引用 .NET 中的对象，方便实现 .NET 的 ref,out 输出参数。
*/

Markdown 格式