aardio 范例: 简单 HTML 解析

```
//简单 HTML 解析

/*
string.html基于string.xml，更适合用来解析HTML,
对于HTML中的笔误等努力尝试修正为正确的结构(例如属性值为空或没放在引号中,标记忘记关闭,
忘记写开始标识不配对,或大小写首尾不匹配 - 关于大小写会首先尝试严格配对,配对不成功会检测是否笔误并进行修正 )

注意此支持库的作用是简单解析,校验HTML错误等不是此支持库的目标,
所以只会尽可能的解析出能解析的结果,尽可能宽容错误写法并试图自动修正。如果需要比较严谨的XML解析器 - 请使用标准库中的 web.mshtml
*/
import string.html;
import console;

var html =  /*
<!doctype html>
<html>
<head></head>
<body>
<table id="container">
<tr><td rowspan="1" class="tab_time tab_time102540630">06:30</td></tr>
</table>
</body>
*/

var htmlDoc = string.html( html )

/*
上面的htmlDoc表示根节点,
htmlDoc包含一个所有子节点的数组,
例如 htmlDoc[1] 表示第一个子节点.

注意string.xml里返回的所有节点对象都是数组形式存在,
例如这里即使只有一个body节点，也要写htmlDoc.body[1]

以 htmlDoc.body[1] 为例，其所有子节点可以用数组索引如下访问
htmlDoc.body[1][1] 第一个子节点
htmlDoc.body[1][2] 第二个子节点

如果要找htmlDoc.body[1]下的div节点，就要这样写
htmlDoc.body[1].div[1] 第一个div子节点
htmlDoc.body[1].div[2] 第二个个div子节点
*/

//查询所有tr节点，在参数中用一个表指定需要查找匹配的节点属性值（支持模式匹配）
var trs = htmlDoc.queryEles( tagName = "tr" );

//遍历所有tr节点，在参数中用一个表指定需要查找匹配的节点属性值（支持模式匹配）
for tr in htmlDoc.eachQuery( tagName = "tr" ){
    //遍历tr节点下的所有td节点
    for i,td in table.eachIndex(tr.td ){
        console.log(td)
    }
}

console.more(1);

var body = htmlDoc.queryEles( tagName = "body");
for(index,tagName,childCount,xNode in body[1].eachChild() ){
    console.log( index,tagName,childCount,xNode.outerXml() )
}

console.log("可以使用id或name属性直接获取节点");
console.log( htmlDoc.getEle("container").outerXml(true) );

//自文档中移除节点
htmlDoc.getEle("container").remove();

//添加节点
body[1].pushElement(tagName = "span";style="font-size:12px").pushElement(
    text = "这是文本节点"
)

//也可以直接写HTML添加节点
body[1].pushXml(`<span class="test">直接写HTML也可以</span>`)
```

```
//在body最后面添加一个超链接
body[1].pushElement(
    tagName = "a";
    href = "http://www.aardio.com";
)

//修改节点属性
body[1].a[1].href ="http://bbs.aardio.com"

//添加文本节点
body[1].a[1].pushElement(
    text = "这是一个超链接"
)

//根据innerText查找节点
var ele = body[1].queryEle(
    tagName = "a";
    innerText = "这是一个超链接"
);

console.log( ele.outerXml(true) )

console.clearScreen();
console.log( htmlDoc.outerXml(true) )
/**
string.html里一个节点对象是一个表,
这个节点对象如果有tagName属性表示标记名 - 那他就是一个普通节点。
如果没有tagName而是用text或cdata属性 - 那就说明他是一个text文本节点或者是cdata文本节点。
=======================================
普通节点使用 tagName 属性表示XML标记名,
"tagName"属于保留字,其他属性使用此名字会被自动忽略,
根节点无tagName,注意这里的根节点指的是文档里的XML根节点的父节点,也就是总是虚拟出一个空的根节点。

文本节点使用 text 属性表示文本,无tagName,无其他属性
CDATA节点使用 cdata 属性表示数据, 无tagName,无其他属性

注释节点被自动忽略不会存为节点对象
xml声明节点的tagName为"?xml"
**/


console.pause()
```

Markdown 格式