Widget Developer’s Reference

目录

前言

如何使用本指南

版本历史

本书中的信息就科泰世纪而言是最新的。最后一次对内容进行修改的时间是 2008 年 X月XX 日。可从客户支持站点处获得对本书中内容的更正或更新。请参阅“技术支持”

文档编排

本手册的内容安排如下:
第一章 开发语言参考， 介绍XmlGlue支持的XML标签, JavaScript对象及全局函数。
第二章 其他 ， 介绍Widget控件的风格。

格式规范

以下格式规范和命令行语法规范将在本书中使用。

说明	涵义	示例
Verdana/宋体 字体：		
斜体字	参考书目	*Elastos IDE User's Guide*
	需强调的文字	… *仅有的编译器*…
引用	窗口或对话框中的提示信息	“本条信息已保存，请不要重复保存！”
带双右尖括号，且被方括号【】 括起的文字	菜单路径	【文件】>>【保存】
粗体字	属性、事件或方法的名称	*window*.**focus** <**pushButton** **id**=”button1” **text**=”Send”>
被方括号【】 括起的文字	菜单选项	选择按钮
	窗口名称	【Output】窗口
	对话框名称	【Settings】对话框
	对话框按钮名称	单击 【OK】
	选项卡名称	单击【属性】选项卡
尖括号< > 括起的文字	键盘上的按键	按<Enter>， <F1>

说明	涵义	示例
Courier New 字体：		
常规Courier New	源代码示例	#define START
	文件名	autoexec.bat
	文件路径	c:\mcc18\h
	命令行选项	-Opa+, -Opa-
	位值	0, 1
	常数	0xFF, 'A'
斜体Courier New	可变参数	run *filename1*，其中 *filename1*可以是任一有效文件名
方括号[ ]	可选参数	
竖线 |	表示或者选择	[option1] | [option2] 在此例中，您必须在 option1 和 option2 中选择一个。
省略号 …	代替重复文字	var_name [, var_name...]
	表示由用户提供的代码	void main (void) { ... }

推荐读物

最新下载文档 可以从科泰Download Center下载文档，网址为：
www.kortide.com.cn

技术支持

科泰世纪的技术支持旨在及时向您提供有关使用科泰软件产品时困难的准确解决方法。
用户可通过以下渠道获得帮助：

- 网站: www.kortide.com.cn
- 电话： 86 21 5131-4260
- 传真： 86 21 5131-4261
- E_Mail: support@kortide.com.cn

第一章 开发语言参考

Figure1-1 描述 Widget 各元素之间的嵌套关系：

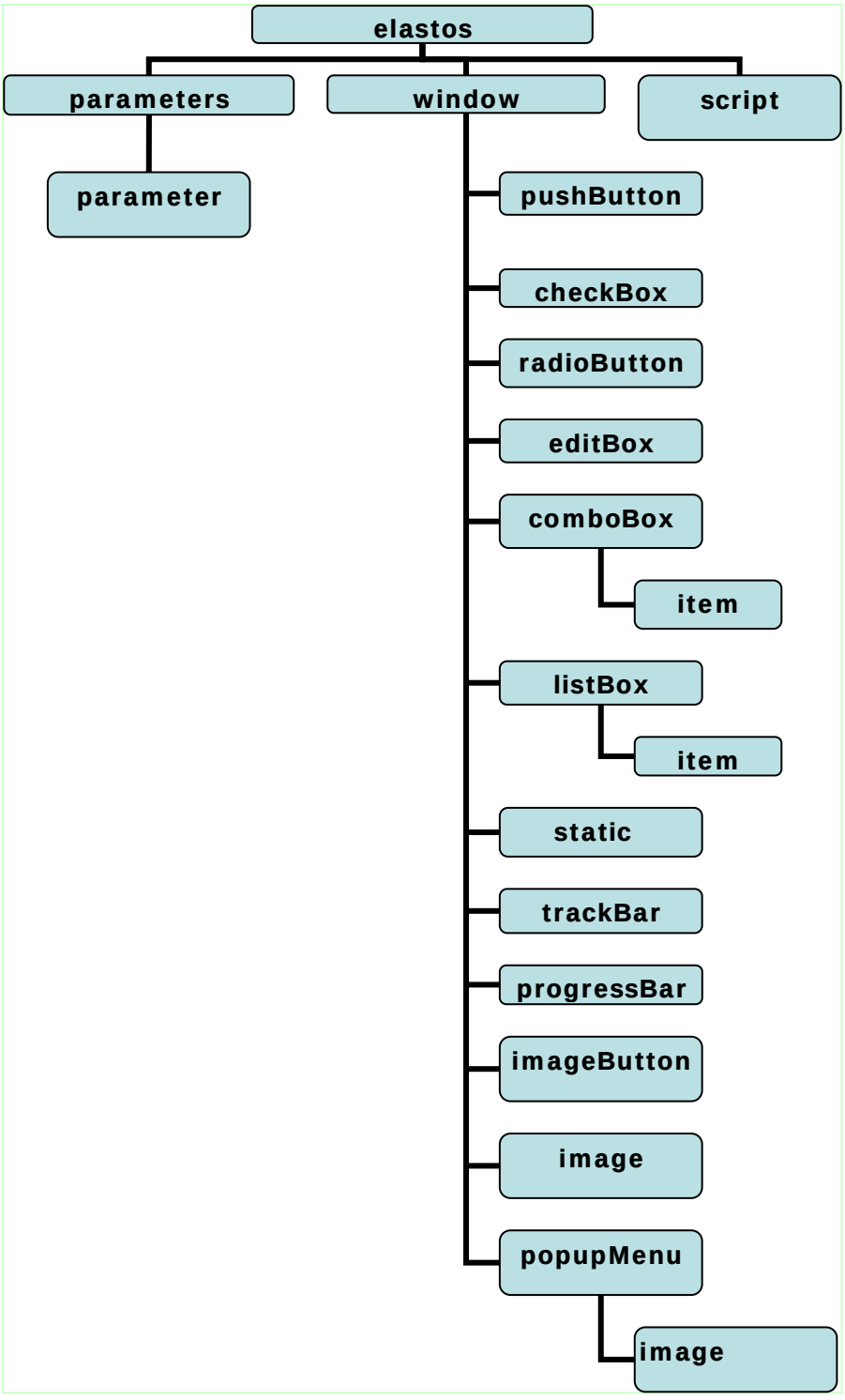

Figure 1-1 标签关系图

1.1 通用属性

通用属性和事件适用于所有 Widget 元素，下表描述通用属性（事件）和方法。

Table 1-1 通用属性表

Attribute	Value	Usage	Example	Description
id	String	XML	`<x:elastos xmlns="http://www.elastos.com/XmlGlue/Widgets" xmlns:x="http://www.elastos.com/XmlGlue/2.1"> <window text="widget" left="0" top="0" width="240" height="320"> <pushButton x:id="myPushButton" left="10" top="270" width="100" height="60"/> </window> </x:elastos>`	此属性用于设置元素的唯一标识. 由于标识将用于程序代码中，它不可以包含任何空格或非 ASCII 字符。一经指定，对象名称将无法改变。
style	string	XML	`<trackBar style="0x01">`	此属性用于设置标签所使用的风格。风格参见控件风格
left	int	XML	`<pushButton top="10" left="10" width="120" height="40"/>`	设置控件内部的左边与它的容器的左边之间的距离。
top	int	XML	`<pushButton top="10" left="10" width="120" height="40"/>`	设置控件的内顶部和它的容器的顶边之间的距离。
width	int	XML	`<pushButton top="10" left="10" width="120" height="40"/>`	设置空间的水平尺寸。
height	int	XML	`<pushButton top="10" left="10" width="120" height="40"/>`	设置对象的垂直尺寸。
text	String	XML	`<pushButton text="OK"/> <window text="widget" left="0" top="0" width="240" height="320"/>`	设置元素的标题显示文字。
		JavaScript	`myPushButton.text = "OK"; print(myPushButton.text);`	

Attribute	Value	Usage	Example	Description
enabled	boolean	XML	`<pushButton enabled="false" />`	设置控件是否响应用户交互作操作。 true:控件响应用户操作； false：控件忽略用户操作。
		JavaScript	`myPushButton.enabled = false;`	
visible	boolean	XML	`<pushButton visible="true"/>`	设置控件是否可见。 **true**:控件可见 **false**:不可见
		JavaScript	`myPushButton.visible = true;`	
onFocus		XML	`<pushButton onFocus="if(getFocus) print('get focus'); else print('lost focus');"/>`	控件获得焦点时，产生**onFocus**事件。 此事件返回一个布尔型参数 getFocus： getFocus=**true** 指示控件获得焦点； getFocus=**false** 指示控件失去焦点。

		JavaScript	`function OnFocus(src, getFocus) {`
			`if (src == myPushButton) {`
			`  if(getFocus)`
			`  print("get focus");`
			`  else`
			`  print("lost focus");`
			`}`
			`}`
			`myPushButton.onFocus = OnFocus;`	

1.2 通用方法

Method	Syntax	Example	Description
MoveTo	**MoveTo**(*newLeft*:number, *newTop*:number) : Void	**Example 1**: `myWindow.MoveTo(150, 150);`	此方法用于移动控件的位置。即：控件的左上角将其由当前位置移动到由坐标 (*newLeft, newTop)* 指定的位置。

		Example 2:
		`var newleft = 20; var newtop = 20;`
		`myPushButton.MoveTo(newleft, newtop);`	

Method	Syntax	Example	Description

Resize	**Resize**(*width*:number,*height*:number):Void	**Example 1**:
		myWindow. Resize(20, 30);
		
		Example 2:
		var newwidth = 120;
		var newheight = 60;
		myPushButton.Resize(newwidth, newheight);
			此方法用于调整控件尺寸。
			参数***width***:调整后控件的水平尺寸;
			参数***height***：调整后控件的垂直尺寸。

GetCurrentPosition	**GetCurrentPosition**():JSObject	var buttonPosition = myPushButton.GetCurrentSize();
		print("button left = " + buttonPosition.x);
		print("button top = " + buttonPosition.y);
			此方法用于获取控件位置，即控件左上角的x,y坐标（相对于屏幕左上角作为坐标原点）。
			方法返回值是一个结构体；此结构体（**position**）有2个属性：**position.x**指示控件位置左上角x坐标,
			position.y指示控件位置左上角y坐标.

GetCurrentSize	**GetCurrentPosition**():JSObject	var buttonSize = myPushButton.GetCurrentSize();
		print("button width = " + buttonSize.width);
		print("button height = " + buttonSize.height);
			此方法用于获取控件的大小，即控件的宽和高。
			方法返回值是一个结构体；此结构体（**size**）有2个属性：
			size.width指示控件宽度；
			size.height指示控件高度.

SetEnabled	**SetEnabled**(*enabled*:Boolean) : Void	**Example 1**:
		myWindow.SetEnabled(false);
		
		Example 2:
		myPushButton.SetEnabled(true);
			此方法设置控件是否对用户交互做出响应。
			参数***enabled***为布尔值。

IsEnabled	**IsEnabled**():Boolean	Value=myWindow.IsEnabled（）；
		if (myPushButton.IsEndbled()) {
		...
		}
			此方法用于确定控件是否对用户交互做出响应。
			方法返回"true",说明控件对用户交互做出响应；返回"false", ,则控件对用户交互不响应.

SetVisible	**SetVisible**(*visible*:Boolean):Void	myWindow.SetVisible(false); myPushButton.SetVisibl	此方法设置控件是否可见。 参数***visible***为布尔

Method	Syntax	Example	Description
		e(true);	值。
IsVisible	**IsVisible**():Boolean	if (myPushButton.IsVisible()) { ... }	此方法用于确定控件是否可见。 方法返回"true"，说明控件可见；返回"false"，则控件不可见。
SetText	**SetText**(*text*:String):Void	**Example 1**: myWindow.SetText("My first Widget");	此方法用于添加控件的标题文字。 参数*text* 为字符串.
		Example 2: myPushButton.SetText("ok");	
GetText	**GetText()**:String	Value=myWindow.GetText(); print(myPushButton.GetText());	此方法用于获取控件的标题内容。

1.3 标签及对象

<elastos> 标签

命名空间URI （namespaceURI）
http://www.elastos.com/XmlGlue/2.1

说明 （Description）
<elastos> 是根元素，可以包含任何元素。
widget文件必须包括一个<elastos>元素。<elastos>元素没有对应的图形表示，用于确定您的widget文件是一个合法的XML文件。

示例（Example）

```
<?xml version="1.0" encoding="utf-8" ?>
< elastos xmlns=" http://www.elastos.com/XmlGlue/2.1"
xmlns:x="http://www.elastos.com/XmlGlue/Widgets">
.......
</elastos>
```

父元素 （Parent Element）
无

属性 （Attribute）

无

<script> 标签

命名空间URI （namespaceURI）
http://www.elastos.com/XmlGlue/2.1

说明 （Description）
<script>是包含脚本的XML元素，用来定义Widget脚本部件运行。
如果在脚本部件文件顶端包含了〈？XML？〉声明，从而启用了XML有效性检验。可将实际的脚本包装在<![CDATA[.....]]>节中。

注：除非声明了〈？XML？〉，否则不要包含CDATA节。

示例（Example）

示例1

```
<?xml version="1.0" encoding="utf-8" ?>
< elastos xmlns=" http://www.elastos.com/XmlGlue/2.1"
xmlns:x="http://www.elastos.com/XmlGlue/Widgets">
        <x:window text="widget" left="20" top="20" width="150" height="190">
            ......
        </x:window>
        < script src="validate.js" Language=”JavaScript”>
        </script>
……
</elastos>
```

示例2

```
<?xml version="1.0" encoding="utf-8" ?>
< elastos xmlns=" http://www.elastos.com/XmlGlue/2.1"
xmlns:x="http://www.elastos.com/XmlGlue/Widgets">
        <x:window text="widget" left="20" top="20" width="150" height="190">
            ......
        </x:window>
        <script>
            <![CDATA[
            function OnSelect1()
```

```
            {
            Window.KillTimer(1);
            Window.Close();
            }
          function OnTimer(id)
            {
            if (id == 1) {
            i = i % 17 + 1;
            img.image = imgs[i];
            }
            }
            button2 = new PushButton("test2", 0, 10 ,60 ,120, 40);
            button2.onSelect = OnSelect1;
            Window.SetTimer(1, 80);
            ]]>
        </ script>
……
</ elastos>
```

父元素 （Parent Element）

elastos

属性 （Attribute）

Attribute	Value	Requirement	Description
language	String	Optional	在脚本部件文件中使用的脚本语言的名称。
src	string	Optional	包含在<script>块中的脚本文件名。

<parameters>标签

命名空间URI （namespaceURI）

http://www.elastos.com/XmlGlue/2.1

说明 （Description）

<parameters> 标签中包含一个或多个**<parameter>**节点。

示例（**Example**）

```
<?xml version="1.0" encoding="utf-8" ?>
<x:elastos xmlns="http://www.elastos.com/XmlGlue/Widgets"
xmlns:x="http://www.elastos.com/XmlGlue/2.1">
        <x:parameters>
            <x:parameter name="ButtonText" required="true" />
        </x:parameters>
        <window text="widget" left="0" top="10" width="240" height="280"
onTimer="OnTimer(timerId)">
            <pushButton x:id="button1" text="${ButtonText}" left="20" top="20"
width="200" height="40" onSelect="OnSelect1();" />
        </window>
        <x:script>
            ……
        </x:script>
……
</x:elastos>
```

父元素 （Parent Element）
<elastos>

属性 （Attributes/Properties）
无

<parameter>标签

命名空间URI （namespaceURI）
http://www.elastos.com/XmlGlue/2.1

说明 （Description）
<parameter>用于指定当前xml文件可以接受的参数。
如需指定参数，应首先使用 **<parameters>** 标签定义参数群组，然后再在群组中设定要的每个参数设置的属性。

示例（Example）

```
<?xml version="1.0" encoding="utf-8" ?>
```

```
< elastos xmlns=" http://www.elastos.com/XmlGlue/2.1"
xmlns:x="http://www.elastos.com/XmlGlue/Widgets">
        < parameters>
            < parameter name="parm1" required="true"/>
            < parameter name="parm2" required="false" defaultValue="hello"/>
        </ parameters>
        <x:window text="widget" left="20" top="20" width="150" height="190">
            <pushButton x:id="button1" text="${parm2}" left="10" top="10" width="120"
height="40"/>
            <imageButton x:id="imgButton" left="10" top="60" width="120" height="108"
upImage="${parm1}" downImage="dancer5.png"/>
        </ x:window>
        < script>
        ...
        </script>
</elastos>
        Application.LoadWindow("XXX.xml?parm1=value1");
        Application.LoadWindow("XXX.xml?parm1=value1&parm2=value2");
```

父元素 （**Parent Element**）

<Parameters>

属性 （**Attributes/Properties**）

Attribute /Property	Value	Usage	Example	Description
name	string	XML	`<x:parameter name="parm1" required="true"/>`	此属性用于指定参数的名称。 **注意：该属性是必须的。**
required	boolean	XML	`<x:parameter name="parm1" required="true"/>`	此属性说明用户在调用xml时是否必须传入此参数。 **注意：该属性是必须的。**
defaultValue	string	XML	`<x:parameter name="parm2" required="false" defaultValue="hello`	此属性表示一个缺省值。 **注意**：当 **required**=”fals

			"/>	e"时，必须指定**defaultValue**属性。

<window> 标签 | window 对象

命名空间URI （namespaceURI）
http://www.elastos.com/XmlGlue/Widgets

说明 （Description）
窗口对象可作为其它对象的容器。可以在 XML 中将其它对象置于框体之内，也可以使用 JavaScript 来将其它对象放到框体之内。框体移动之后，所有子视图也会移动。

示例（Example）

下面的代码示例了一个完整的widget文件。在代码中，我们创建了名为**widget**的窗口，窗口包含**Quit**按钮的**PushBotton** 控件和**ImageButton** 控件；单击**Quit** 即将关闭窗口。

```
<?xml version="1.0" encoding="utf-8" ?>
<x:elastos xmlns="http://www.elastos.com/XmlGlue/Widgets"
xmlns:x="http://www.elastos.com/XmlGlue/2.1">
        <window text="widget" left="20" top="20" width="150" height="190">
            <pushButton x:id="button1" text="Quit" left="10" top="10" width="120"
height="40" onSelect="OnSelect1();" />
            <imageButton x:id="imgButton" left="10" top="60" width="120" height="108"
upImage="dancer1.png" downImage="dancer5.png" />
        </window>
        <x:script>
            <![CDATA[
                            function OnSelect1()
                            {
                            Window.Close();
                            }
            ]]>
        </x:script>
</x:elastos>
```

父元素 （Parent Element）

<elastos>

属性 （Attributes/Properties）

Attribute /Property	Value	Usage	Example	Description
onTimer		XML	`<window text="widget" left="0" top="0" width="240" height="320" onTimer= "if (id == 1) { ... }"/>`	当**setTimer**函数指定的时间到期后，产生**onTimer**事件，用户注册的事件处理函数会被调用。 **注**：本事件应包含一个参数*id* **注**：*id*由**setTimer**函数设置。
		JavaScript	`function OnTimer (src, id) { if (src == Window && id == 1) {... } } Window.onTimer = OnTimer;`	
onClose		XML	`<Window text="widget" left="0" top="0" width="240" height="300" onClose =" print(&quot;window closed!&quot;)"/>`	窗口被关闭时，会触发**onClose**事件。 本事件无返回参数。
		JavaScript	`function OnClose(src) { if (src == Window) print("window close!"); return 1; } Window.onClose = OnClose;`	

其他属性 见 通用属性

方法

Close
Show
SetTimer
KillTimer
SetContextMenu
GetGrafix
见 通用方法

Close

描述（Description）

关闭窗口。

语法（Syntax）

```
window.Close () :Void
```

注释（Remark）

无

示例 (Example)

示例1

```
<?xml version="1.0" encoding="utf-8" ?>
< elastos xmlns=" http://www.elastos.com/XmlGlue/2.1"
xmlns:x="http://www.elastos.com/XmlGlue/Widgets">
        <x:window text="widget" left="20" top="20" width="150" height="190">
            .......
        </x:window>
        < script >
            <![CDATA[
            function OnSelect1()
            {
            window.Close();
            }
            button = new PushButton("Click this button and window.close() is called.",
0, 10 ,60 ,120, 40);
            button.onSelect = OnSelect1;
            ]]>
        </ script>
        ……
</elastos>
```

示例2

```
<?xml version="1.0" encoding="utf-8" ?>
```

```
<x:elastos xmlns="http://www.elastos.com/XmlGlue/Widgets"
xmlns:x="http://www.elastos.com/XmlGlue/2.1">
        <window text="widget" left="0" top="0" width="240" height="320"
onTimer="OnTimer(timerId)">
            <pushButton x:id="button1" text="quit" left="120" top="250" width="100"
height="30" onSelect="Window.Close()" />
            <pushButton x:id="button2" text="Draw" left="10" top="250" width="100"
height="30" onSelect="testDraw();" />
        </window>
        <x:script>
            ……
        </x:script>
</x:elastos>
```

Show

描述（Description）

显示当前窗口。

语法（Syntax）

```
        window.Show():Void
```

注释（Remark）

示例 (Example)

在下面的代码示例中：当您单击**button1**，将调用函数**OnSelect1. 在OnSelect1**函数中，**Application.LoadWindow** 方法被调用，完成启动**main.xml**窗口，然后使用**Show**方法去显示**main.xml**窗口。

```
<? xml version="1.0" encoding="utf-8" ?>
```

```
<x:elastos xmlns="http://www.elastos.com/XmlGlue/Widgets"
xmlns:x="http://www.elastos.com/XmlGlue/2.1">
        <window text="widget" left="0" top="10" width="240" height="280"
onTimer="OnTimer(timerId)">
            ……
        </window>
        <x:script>
        <![CDATA[
            function OnSelect1()
            {
            var newWnd = Application.LoadWindow("main.xml");
            newWnd.Show();
            }
            function OnSelect2()
            {
            Application.Quit();
            }
            button1 = new PushButton("NewWnd", 0, 130 ,200 ,100, 40);
            button1.onSelect = OnSelect1;
            button2 = new PushButton("Quit", 0, 10 ,200 ,100, 40);
            button2.onSelect = OnSelect2;
            ]]>
            </x:script>
</x:elastos>
```

SetTimer

描述（Description）

设置计时器。

语法（Syntax）

```
window. SetTimer(id, timeout):Void
```

参数*id* 整数，指定计时器序号；
参数*timeout*整数，指定时间间隔，单位毫秒。

注释（Remark）

示例 (Example)

在下面的代码示例中：窗口中定义了名为**button1**的**pushButton**控件；在JavaScript代码中，使用**SetTimer**方法设置一个计时器（序号：1；间隔：80ms）。

当您单击**button1**，将调用函数**OnSelect1**；**OnSelect1**完成两件事情：1．调用**Close**方法关闭窗口，2．调用**KillTimer**方法终止计时器。

```
<?xml version="1.0" encoding="utf-8" ?>
<x:elastos xmlns="http://www.elastos.com/XmlGlue/Widgets"
xmlns:x="http://www.elastos.com/XmlGlue/2.1">
        <x:parameters>
            <x:parameter name="ButtonText" required="true" />
        </x:parameters>
        <window text="widget" left="0" top="10" width="240" height="280"
onTimer="OnTimer(timerId)">
            <pushButton x:id="button1" text="${ButtonText}" left="20" top="20"
width="200" height="40" onSelect="OnSelect1();" />
        </window>
        <x:script>
        <![CDATA[
            function OnSelect1()
            {
            Window.KillTimer(1);
            Window.Close();
            }
            function OnTimer(id)
            {
            print("id == " + id);
            if (id == 1)      {
```

```
                                ……
                                }
            }
            Window.SetTimer(1, 80);
            ]]>
            </x:script>
</x:elastos>
```

KillTimer

描述（Description）
终止计时器。

语法（Syntax）

```
        window.KillTimer(id) :Void
```

参数 *id*为将被终止计时器的序号

注释（Remark）

示例 (Example)

```
        window.KillTimer(1)
```

SetContextMenu

描述（Description）
此方法用于把一个弹出式菜单指定为窗口（window）的菜单，当用户在窗口上长按触笔的时候自动显示.

语法（Syntax）

```
        window.SetContextMenu(popupMenu) :Void
```

参数 *popupMenu* 为弹出式菜单（PopupMenu）对象。

注释（Remark）
用户可以在XML标签里定义一个弹出式菜单，它的父亲必须是窗口（window），并且一个窗口不可以有一个以上的弹出式菜单。

示例 (Example)

```
var popup = new PopupMenu();
popup.AddStringItem(“ItemOne”, 123);
window.SetContextMenu(popup)
```

GetGrafix

描述（Description）

获得窗口的绘图对象。用户可以使用这个绘图对象的方法在窗口上绘制图形。

语法（Syntax）

```
window.GetGrafix():JSObject
```

返回值（Reture Value）

XGrafix对象

注释（Remark）

无

示例 (Example)

```
var Grafix= window.GetGrafix();
```

<pushButton> 标签 | PushButton 对象

命名空间URI （namespaceURI）

http://www.elastos.com/XmlGlue/Widgets

说明 （Description）

pushButton 控件即常见的按钮；允许用户通过单击来执行操作。

示例（Example）

```
<?xml version="1.0" encoding="utf-8" ?>
< elastos xmlns=" http://www.elastos.com/XmlGlue/2.1"
xmlns:x="http://www.elastos.com/XmlGlue/Widgets">
```

```
        <x:window text="widget" left="20" top="20" width="150" height="190">
        <pushButton id="myPushButton" text="Click" left="10" top="10" width="120"
height="40"/>
        </x:window>
……
</ elastos>
```

JavaScript中：

```
<?xml version="1.0" encoding="utf-8" ?>
< elastos xmlns=" http://www.elastos.com/XmlGlue/2.1"
xmlns:x="http://www.elastos.com/XmlGlue/Widgets">
        <x:window text="widget" left="20" top="20" width="150" height="190">
.......
        < script >
            <![CDATA[
            function OnSelect()
            {
            Window.KillTimer(1);
            Window.Close();
        }
           function OnTimer(id)
            {
            if (id == 1) {
            i = i % 17 + 1;
            img.image = imgs[i];
        }
        }
            var myButton = new PushButton("Click", 0, 10, 10, 120, 40);
            myButton.onSelect = OnSelect1;
            Window.SetTimer(1, 80);
             ]]>
        </ script>
</ elastos>
```

父元素 （**Parent Element**）

<window>

属性 （**Attributes/Properties**）

Attribute /Property	Value	Usage	Example	Description
onSelect		XML	`<pushButton x:id="button1" text="quit" left="180" top="250" width="50" height="40" onSelect="Window.Close()" />`	当用户单击按钮时，产生**onSelect**事件。同时**onSelect**指定的事件处理程序将被执行。
		JavaScript	`myButton.onSelect= Onselect()`	

其他属性 见 通用属性

构造函数

PushButton

描述（**Description**）

此方法创建一个按钮。

语法（**Syntax**）

```
var button = new PushButton (text, style, left, top, width, height);
```

text 为控件标题；该参数不可缺省。

style 为控件风格，可用的样式包括控件通用样式以及按钮通用样式，参见控件风格。该参数不可缺省。

left 整数，指定按钮的左边到它的父亲容器的左边的距离，以屏幕像素为单位。该参数不可缺省。

top 整数，指定按钮的顶边到它的父亲容器的顶边的距离，以屏幕像素为单位。该参数不可缺省。

width 整数，指定按钮的水平尺寸，以屏幕像素为单位。该参数不可缺省。

height 整数，指定按钮的垂直尺寸，以屏幕像素为单位。该参数不可缺省。

注释（**Remark**）

示例 **(Example)**

```
<?xml version="1.0" encoding="utf-8" ?>
<x:elastos xmlns="http://www.elastos.com/XmlGlue/Widgets"
xmlns:x="http://www.elastos.com/XmlGlue/2.1">
        <x:parameters>
            <x:parameter name="ButtonText" required="true" />
```

```
        </x:parameters>
        <window text="widget" left="0" top="10" width="240" height="280"
onTimer="OnTimer(timerId)">
            <pushButton x:id="button1" text="${ButtonText}" left="20" top="20"
width="200" height="40" onSelect="OnSelect1();" />
        </window>
        <x:script>
        <![CDATA[
            button2 = new PushButton("NewWnd", 0, 130 ,200 ,100, 40);
            button2.onSelect = OnSelect2;
            button3 = new PushButton("Quit", 0, 10 ,200 ,100, 40);
            button3.onSelect = OnSelect3;
            Window.SetTimer(1, 80);
            Grafix = Window.GetGrafix();
            Grafix.Rect(50, 90, 120, 108);
            Grafix.Clip();
            Grafix.Translate(50, 90);
            function OnSelect1()
            {
            Window.KillTimer(1);
            Window.Close();
            }
            function OnSelect2()
            {
            button1.enabled = true;
            var newWnd = Application.LoadWindow("main.xml");
            newWnd.Show();
            }
            function OnSelect3()
            {
            Window.KillTimer(1);
            Application.Quit();
            }
```

```
            imgs = new Array(18);
            for (i = 1; i < 18; i++) {
            imgs[i] = new Image("dancer" + i + ".png");
            }
            var i = 0;
            function OnTimer(id)
            {
            print("id == " + id);
            if (id == 1)      {
                              i = i % 17 + 1;
                              Grafix.ClearRect(0, 0, 120, 108);
                              Grafix.DrawImage(imgs[i], 0, 0);
                              Grafix.Update();
                              }
            }
            ]]>
            </x:script>
</x:elastos>
```

方法

SetFocus

其他方法 见 通用方法

SetFocus()

描述（Description）

使控件获得焦点。

语法（Syntax）

```
        button.SetFocus()
```

注释（Remark）

示例 (Example)

```
        myButton.SetFocus()
```

<checkBox>标签 | CheckBox 对象

命名空间URI （namespaceURI）
http://www.elastos.com/XmlGlue/Widgets

说明 （Description）

checkBox（复选框控件） 指示某个特定条件是处于打开状态还是处于关闭状态。
示例（Example）

下面的代码示例创建了一个 checkBox 并对它进行了初始化，为它赋予切换按钮的外观.

```
<?xml version="1.0" encoding="utf-8" ?>
< elastos xmlns=" http://www.elastos.com/XmlGlue/2.1"
xmlns:x="http://www.elastos.com/XmlGlue/Widgets">
        <x:window text="widget" left="20" top="20" width="150" height="190">
            <checkBox x:id="myCheckBox" text="CheckBox" style=”0x1” left="10"
top="10" width="120" height="40" checked = “true” />
        </x:window>
……
</ elastos>
```

JavaScript中：

```
        var myCheckBox = new CheckBox("CheckBox",CheckBox.DROPDOWNLIST, 10, 120, 40,
0);
        function Checked(src)
        {
            if (src == myCheckBox) print("Checked!");
        }
        myCheckBox.onSelect = Checked;
        myCheckBox.checked = true;
```

父元素 （Parent Element）
<window>

属性 （Attributes/Properties）

Attribute /Property	Value	Usage	Example	Description
onSelect		XML	<checkBox ... onSelect="print("Checked!")"/>	当用户单击按钮时，产生**onSelect**事件。同时**onSelect**指定的事件处理程序或代码将被执行。 本事件无参数

		JavaScript	function Checked(src) {
			if (src == myCheckBox)
			print("Checked!");
			}
			myCheckBox.onSelect = Checked;	

checked	boolean	XML	<checkBox ... checked="true"/>	获取或设置一个值，该值指示 checkBox 是否处于选中状态
		JavaScript	myCheckBox.checked = true;	

其他属性 见 公共属性

构造函数

CheckBox

描述（Description）

创建一个复选框控件。

语法（Syntax）

```
var button = new CheckBox (text, style, left, top, width, height)
```

text 为控件标题；该参数不可缺省。
style 为控件风格，可用的样式包括控件通用样式以及按钮通用样式，参见控件风格。该参数不可缺省。
left 整数，指定按钮的左边到它的父亲容器的左边的距离，以屏幕像素为单位。该参数不可缺省。
top 整数，指定按钮的顶边到它的父亲容器的顶边的距离，以屏幕像素为单位。该参数不可缺省。
width 整数，指定按钮的水平尺寸，以屏幕像素为单位。该参数不可缺省。
height 整数，指定按钮的垂直尺寸，以屏幕像素为单位。该参数不可缺省

返回值（Return Value）

返回一个复选框对象。

注释（Remark）

示例 (Example)

```
var checkbox = new CheckBox (“Red”,CheckBox.BORDER,200,300,20,30)
```

方法小结

SetChecked
IsChecked
SetFocus
其他方法 见 通用方法
SetChecked

描述（Description）
设置单选（或复选）按钮的选择状态。

语法（Syntax）

```
button.SetChecked(checked)
```

参数*checked*为布尔值。

返回值（Return Value）
无

注释（Remark）

示例 (Example)

```
myCheckBox.SetChecked(true);
```

IsChecked ()

描述（Description）
检查返回单选（或复选）按钮的选择状态。

语法（Syntax）

```
Value = button. IsChecked ( )
```

返回值（Return Value）
布尔值
true：被选中
false：没有选中

注释（Remark）

示例 (Example)

```
Value = myButton. IsChecked（）;
if (myCheckBox.IsChecked()) {
...
};
```

SetFocus()

描述（Description）
使控件获得焦点。

语法（Syntax）

```
object.SetFocus ( )
```

返回值（Return Value）
无

注释（Remark）

示例 (Example)

```
myCheckbox.SetFocus ( )
```

<radioButton> 标签 | RadioButton 对象

命名空间URI （namespaceURI）
http://www.elastos.com/XmlGlue/Widgets

说明 （Description）
checkBox 和 radioButton (单选框控件)拥有一个相似的功能：允许用户从选项列表中进行选择。checkBox 控件允许用户选择一组选项。与之相反，radioButton 控件允许用户从互相排斥的选项中进行选择。

注意，在本版中，一个window之内的radioButton都缺省为处于同一组。

示例（Example）

```
        <radioButton x:id="myRadioButton" text="RadioButton" left="10" top="10"
width="120" height="40"/>
```

JavaScript中：

```
        var myrbt = new RadioButton("RadioButton", 10, 10, 120, 40, 0);
        myrbt.checked = true;
        function Selected(src) {
                if (src == myrbt) print("Selected!");
        }
        myrbt.onSelect = Selected;
```

父元素 （Parent Element）

<window>

属性 （Attributes/Properties）

Attribute /Property	Value	Usage	Example	Description
onSelect		XML	<radioButton ... onSelect="print("Selected!")"/>	当用户单击鼠标按钮时，产生**onSelect**事件。同时**onSelect**指定的事件处理程序或代码将被调用执行。
		JavaScript	function Selected(src) { if (src == myrbt) print("Selected!"); } myrbt.onSelect = Selected;	
checked	boolean	XML	<radioButton ... checked="true"/>	获取或设置一个值，该值指示是否已选中控件。
		JavaScript	myrbt.checked = true;	

其他属性 见 通用属性

构造函数

RadioButton

描述（Description）

创建一个按钮。

语法（Syntax）

```
        var button = new RadioButton (text, style, left, top, width, height);
```

text 为控件标题；该参数不可缺省。
style 为控件风格，可用的样式包括控件通用样式以及按钮通用样式，参见控件风格。该参数不可缺省。
left 整数，指定按钮的左边到它的父亲容器的左边的距离，以屏幕像素为单位。该参数不可缺省。
top 整数，指定按钮的顶边到它的父亲容器的顶边的距离，以屏幕像素为单位。该参数不可缺省。
width 整数，指定按钮的水平尺寸，以屏幕像素为单位。该参数不可缺省。
height 整数，指定按钮的垂直尺寸，以屏幕像素为单位。该参数不可缺省。

返回值（Return Value）
无

示例 (Example)

```
        var radioButton = new RadioButton (“Workday”,
RadioButton.BORDER,200,300,20,30)
```

方法
SetChecked
IsChecked
SetFocus
其他方法 见 通用方法

SetChecked

描述（Description）
设置单选（或复选）按钮的选择状态。

语法（Syntax）

```
        button.SetChecked(checked)
```

参数*checked*为布尔值。

返回值（Return Value）
无

注释（Remark）

示例 (Example)

```
        myRadioButton.SetChecked(true);
```

IsChecked

描述（Description）
检查返回单选（或复选）按钮的选择状态。

语法（Syntax）

```
        Value = button. IsChecked () ;
```

返回值（Return Value）
布尔值
true：被选中
false：没有选中

注释（Remark）

示例 (Example)

```
Value = myRadioButton. IsChecked () ;
if (myRadioButton.IsChecked()) {
...
 };
```

SetFocus()

描述（Description）
使控件获得焦点。

语法（Syntax）

```
       object.SetFocus () ;
```

返回值（Return Value）
无
注释（Remark）

示例 (Example)

```
        myRadioButton.SetFocus();
```

<imageButton>标签 | ImageButton对象

命名空间URI （namespaceURI）
http://www.elastos.com/XmlGlue/Widgets

说明 （Description）
imageButton是一种可以显示图片的按钮。用户可以指定该按钮在按下、抬起以及不可用（disabled）状态所显示的外观图片。

示例（Example）

下面的代码示例创建了一个 imageButton控件。

```
<x:window text="widget" left="20" top="20" width="150" height="190">
        <imageButton id="imgButton" style=”0” left="10" top="10" width="120"
height="40" upImage="dancer1.png" downImage="dancer5.png"
disabledImage="dancer9.png" onSelect="print(&amp;quot;Selected!&amp;quot;)"/>
</x:window>
```

JavaScript中：

```
        var myImgBtn = new imageButton (10, 120, 40, 0, “xxx.png”, “xx.png”, null);
        function OnMySelection(src) {
                if (src == myImgBtn) print("Checked!");
        }
        myImgBtn.onSelect = OnMySelection;
```

父元素 （Parent Element）
<window>

属性 （Attributes/Properties）

Attribute /Property	Value	Usage	Example	Description

Attribute /Property	Value	Usage	Example	Description
upImage	string	XML	`<imageButton ... upImage="XXX.png"/>`	设置按钮抬起时显示的图片
downImage	string	XML	`<imageButton ... downImage="XXX.png"/>`	设置按钮按下时显示的图片
disabledImage	string	XML	`<imageButton ... disabledImage="XXX.png"/>`	设置按钮不可用时显示的图片
onSelect		XML	`<imageButton ... onSelect="print(&quot;Selected!&quot;)"/>`	当用户单击按钮时，产生 **onSelect** 事件。同时 **onSelect** 指定的事件处理程序或代码将被调用执行。
		JavaScript	`function OnMySelection(src) { if (src == myImgBtn) print("Checked!"); } myImgBtn.onSelect = OnMySelection;`	

其他属性 见 通用属性

构造函数

ImageButton

描述（Description）

创建一个图片按钮控件。

语法（Syntax）

```
var imgButton = new ImageButton(style, left, top, width, height, upImage, downImage, disabledImage)
```

style 控件样式。可用的样式包括控件通用样式以及按钮通用样式，参见控件风格。该参数不可缺省。
left 整数，指定按钮的左边到它的父亲容器的左边的距离，以屏幕像素为单位。该参数不可缺省。
top 整数，指定按钮的顶边到它的父亲容器的顶边的距离，以屏幕像素为单位。该参数不可缺省。
width 整数，指定按钮的水平尺寸，以屏幕像素为单位。该参数不可缺省。
height 整数，指定按钮的垂直尺寸，以屏幕像素为单位。该参数不可缺省。
upImage 字符串，指定按钮未按下时所显示的图片文件位置。该参数不可缺省。
downImage 字符串，指定按钮按下时所显示的图片文件位置。该参数不可缺省。
disabledImage 字符串，指定按钮不可用（disabled）时所显示的图片文件位置。该参数可以缺省。如果缺省该参数，按钮在不可用时将显示未按下的图片。

注意：
*disabledImage*可以缺省；在缺省状态下当控件处于不可用状态（disabled状态）时将显示*upImage*作为其外观。其他两个外观图片不可以缺省。

返回值（Return Value）
返回一个**ImageButton**对象。

注释（Remark）

示例 (Example)

```
var imgButton = new ImageButton (10, 120, 40, 0, “xxx.png”, “xx.png”, null);
```

方法小结
SetFocus
其他方法 见 通用方法

SetFocus()

描述（Description）
使控件获得焦点。

语法（Syntax）

```
object.SetFocus（）;
```

返回值（Return Value）
无
注释（Remark）

示例 (Example)

```
myImgbutton.SetFocus();
```

<menu>标签 | PopupMenu 对象

命名空间URI （namespaceURI）
http://www.elastos.com/XmlGlue/Widgets

说明 （Description）

PopupMenu控件，即弹出式菜单。用户可以在脚本中创建一个弹出式菜单，并在需要的时候把它显示出来；用户也可以把一个弹出式菜单指定为窗口（window）的菜单，由窗口在需要的时候自动显示（请参考window的SetContextMenu 方法）。用户可以在XML标签里定义一个弹出式菜单，它的父亲必须是窗口（window），并且一个窗口不可以有一个以上的弹出式菜单。

示例（**Example**）

```
<menu onSelect="print(&amp;quot;id = &amp;quot; + id);">
        <item id="12" text="item1"/>
        <separator/>
        <item id="13" enabled="false">item2</item>
        <item text="item3">
                <item id="15" text="item4"/>
                <separator/>
                <item id="17" checked="true">item5</item>
                <separator/>
                <item id="16" text="item6"/>
        </item>
        <item id="18">item7</item>
        <separator/>
        <item id="19" text="item8"/>
</menu>
```

JavaScript中：

```
        var popup = new PopupMenu();
        popup.AddStringItem(“ItemOne”, 123);
        popup.AddSeparator();
        var submenu = new PopupMenu();
        submenu.AddStringItem(“SubItemOne”, 234);
        popup.AddSubMenu(“SubMenu”, submenu);
```

父元素 （**Parent Element**）

<window>

属性 （**Attributes/Properties**）

Attribute /Property	Value	Usage	Example	Description

Attribute /Property	Value	Usage	Example	Description
onSelect		XML	<menu onSelect="print("id = " + id);"> .... </menu>	当用户单击菜单项时，产生**onSelect**事件；同时**onSelect**指定的事件处理程序或代码将被调用执行。

		JavaScript	function OnMySelection(src, id) {
			if (id == 123)
			print("ItemOne is selected.");
			}
			popup.onSelect = OnMySelection;	

其他属性 见 通用属性

构造函数

PopupMenu

描述（Description）

创建一个弹出式菜单。

语法（Syntax）

```
var popup = new PopupMenu();
```

返回值（Return Value）

返回一个弹出式菜单对象。

注释（Remark）

示例 (Example)

```
var popup = new PopupMenu();
popup.AddStringItem(“ItemOne”, 123);
popup.AddSeparator();
var submenu = new PopupMenu();
submenu.AddMenuItem(“SubItemOne”, 234);
popup.AddSubMenu(“SubMenu”, submenu);
```

方法小结

AddStringItem
AddSeparator
AddSubMenu
RemoveItem
CheckItem
CheckRadioItem

EnableItem
Show

AddStringItem

描述（Description）
向菜单末尾追加一个文字菜单项。

语法（Syntax）

```
popup.AddStringItem（itemText, itemId）;
```

itemText 字符串（必需）:用于指定菜单项的文字。
itemId 整数（必需）:用于指定菜单项的Id。

返回值（Return Value）
无

注释（Remark）
用户应当保证一个菜单的所有的菜单项的*itemId*不重复（包括子菜单里的*itemId*），否则当菜单的某一项被选中时，用户在脚本代码里将无法区分实际的选项。

示例 (Example)

```
popup. AddStringItem（”ItemOne”,123）;
```

AddSeparator

描述（Description）
将默认大小的分隔符（一条直线）添加到菜单的末尾。

语法（Syntax）

```
popup. AddSeparator（）;
```

返回值（Return Value）
无

注释（Remark）

示例 (Example)

```
popup. AddSeparator（）;
```

AddSubMenu

描述（Description）
向菜单末尾追加一个子菜单项。

语法（Syntax）

```
popup.AddSubMenu（itemText, submenu）
```

itemText 字符串，必需。用于指定子菜单项的文字。
submenu 菜单对象，必需。用于指定所需添加的子菜单项。

返回值（Return Value）
无

注释（Remark）

示例 (Example)

```
var submenu = new PopupMenu();
submenu.AddStringItem(“hello”, 246);
popup. AddSubMenu（”ItemOne”, submenu）
```

RemoveItem

描述（Description）
移除菜单的某一项。

语法（Syntax）

```
popup.RemoveItem(id)
```

id 整型（必需），用于指定要移除菜单项的位序。
注意：菜单第一项的位序是0，第二项位序是1，以此类推。

返回值（Return Value）
无

注释（Remark）
如果被移除的是子菜单，则该子菜单对象并没有被删除，只是从父菜单上脱离。用户还可以使用该子菜单，包括把它添加到其他菜单上去作为子菜单，或者单独使用。

示例 (Example)

```
menu.RemoveItem(0);     //移除菜单的第一项
```

CheckItem

描述（Description）
在菜单的某一项的文字的左边做一个标记，即打一个勾。

语法（Syntax）

```
popup. CheckItem（id, checked）
```

id 整型(必需)：用于指定要标记的菜单项的Id。
checked 布尔型(必需)：**true**，表明需要标记；**false**，表明取消标记。

返回值（Return Value）
无

注释（Remark）

示例 (Example)

```
menu.CheckItem(123, true);     //标记 Id 为”123”的项
```

CheckRadioItem

描述（Description）
在菜单某几项中的一项的文字的左边做一个标记，即打一个圈；如果这几项中已经有另一项被做过标记，则先清除那个标记。

语法（Syntax）

```
popup.CheckRadioItem（idFirst, idLast, idCheck）
```

*idFirst*整型（必需）：指定要标记的起始菜单项的Id。
*idLast*整型（必需）：指定要标记的最后菜单项的Id。
*idFirst*整型（必需）：指定要标记的菜单项的Id。

返回值（Return Value）
无

注释（Remark）

示例 (Example)

```
menu.AddStringItem(“hello”, 123);
menu.AddStringItem(“Hey”, 246);
menu.AddStringItem(“OK”, 222);
menu.CheckRadioItem(123, 222, 246);          //标记 Id 为 246 的项
```

EnableItem

描述（Description）
设置菜单的某一菜单项为可用或不可用。

语法（Syntax）

```
popup.EnableItem（id, enabled）
```

id 整型（必需）：指定要操作的菜单项的Id。
enabled 布尔值（必需）：指定是否要使该项可用。

返回值（Return Value）
无

注释（Remark）
默认情况下菜单的所有项都是可用的。当某一项不可用时，它将不能接受用户输入。

示例 (Example)

```
menu.EnabledItem(123, false);
```

Show

描述（Description）
显示菜单。

语法（Syntax）

```
popup.Show（x, y）;
```

x 整型（必需）：指定菜单显示位置的水平坐标，相对于其父窗口左上角为原点。
y 整型（必需）：指定菜单显示位置的垂直坐标，相对于其父窗口左上角为原点。

返回值（Return Value）
返回用户所选择的菜单项的ID。如果用户什么也没有选择，返回0。

注释（Remark）
可以通过注册**OnSelect**事件来获取用户的选择。注意：如果想要根据方法的返回值来判断，则需要注意不要把菜单项的Id设置为0。

示例 (Example)

```
menu.Show(100, 100);
```

<comboBox>标签 | ComboBox 对象

命名空间URI （namespaceURI）
http://www.elastos.com/XmlGlue/Widgets

说明 （Description）

comboBox 控件用于在下拉组合框中显示数据。默认情况下，comboBox 控件分两个部分显示：顶部是一个允许用户键入列表项的文本框。第二部分是一个列表框，它显示一个项列表，用户可从中选择一项。

示例（Example）

```
<comboBox x:id="option" left="10" top="270" width="100" height="60">
        <item selected="true" text="test line"/>
        <item text="Image 1"/>
        <item text="Image 2"/>
</comboBox>
```

JavaScript中

```
var option = new ComboBox("", 10, 10, 120, 40, 0);
option.AddString("test line");
option.AddString("Image 1");
option.AddString("Image 2");
option.SetCurSelection(0);
function OnListState(src, listState) {
        if (src == option) print(listState);
}
option.onListState = OnListState;
function OnEditState(src, editState) {
        if (src == option) print(editState);
        }
option.onEditState = OnEditState;
```

父元素 （Parent Element）

<window>

属性 （Attributes/Properties）

Attribute /Property	Value	Usage	Example	Description
onListState		XML	<comboBox ... onListState="print(listState)"> ...	当ComboBox的列表状态发生改变时，触发此事件。 事件返回一个参数

			</comboBox>	listState，其可能值为： ➢ LISTSHOW 表示列表将被显示 ➢ LISTHIDE 表示列表刚被隐藏 ➢ LISTSELELECTIONCHANGE 表示用户选择了列表的另一项

		JavaScript	function OnListState(src, listState) {
			if (src == option)
			print(listState);
			}
			option.onListState = OnListState;	

onEditState		XML	<comboBox ... onEditState="print(editState)"> ... </comboBox>	当ComboBox的列表的文本被修改，引起状态的改变，触发此事件。 此事件返回一个参数editState，其可能值为： ➢ EDITUPDAT 控件文本输入框的内容已改变，即将更新屏幕显示 ➢ EDITCHAGNE 控件文本输入框的文本内容已经改变并且屏幕已更新

		JavaScript	function OnEditState(src, editState) {
			if (src == option)
			print(editState);
			}
			option.onEditState = OnEditState;	

其他属性 见 通用属性

构造函数

ComboBox

描述（Description）

创建一个**ComboBox**控件。

语法（Syntax）

```
ComboxBox combo = new ComboxBox (text, style, left, top, width, height);
```

text 为控件标题；该参数不可缺省。

style 为控件风格，可用的样式参见控件风格和ComboBox控件风格。该参数不可缺省。

left 整数，指定按钮的左边到它的父亲容器的左边的距离，以屏幕像素为单位。该参数不可缺省。

top 整数，指定按钮的顶边到它的父亲容器的顶边的距离，以屏幕像素为单位。该参数不可缺省。

width 整数，指定按钮的水平尺寸，以屏幕像素为单位。该参数不可缺省。

height 整数，指定按钮的垂直尺寸，以屏幕像素为单位。该参数不可缺省。

注释（Remark）

示例 (Example)

```
myCombo = new ComboBox (“product1”,ComboBox. DROPDOWNLIST, 200, 300, 20, 30);
```

方法

AddString
InsertString
DeleteString
GetItemCount
ResetContent
GetString
SetTopIndex
GetTopIndex
SetEditMaxChars
SetListWidth
ShowList
SetCurSelection
ClearSelection
GetCurrentSelection
SetFocus
其他方法 见 通用方法

AddString

描述（Description）

向列表末尾添加一项。

语法（Syntax）

```
        var index=Combo.AddString(str);
```

参数*str*为所添加的项目。

返回值（Return Value）

返回值为集合中项的从零开始的索引。

示例 (Example)

```
        index = myCombo.AddString(“Apple”);
```

InsertString

描述（Description）

向列表中第index项之前插入一项。列表的第一项的索引为0。

语法（Syntax）

```
var index=Combo.InsertString(index, str)
```

参数*str*为所添加的项目。

返回值（Return Value）

返回新添加条目的索引。

示例 (Example)

```
index=myComboBox.InsertString(3,“Apple”);
```

DeleteString

描述（Description）

删除列表中第*index*项。

语法（Syntax）

```
var total = combo. DeleteString(index)
```

参数*index:* （整型）为删除项的索引。

返回值（Return Value）

返回剩余条目的总数。

示例 (Example)

```
count = myCombo. DeleteString(3);
```

GetItemCount

描述（Description）
获取列表中条目的总数。

语法（Syntax）

```
var total = combo. GetItemCount()
```

返回值（Return Value）
返回条目的总数。

示例 (Example)

```
var count == myCombo. GetItemCount()
```

ResetContent

描述（Description）
清空控件列表框的所有条目和文本框的内容。

语法（Syntax）

```
combo.ResetContent()
```

返回值（Return Value）
无

示例 (Example)

```
myCombo.ResetContent();
```

GetString

描述（Description）

获取列表中索引为*index*的条目值。

语法（Syntax）

```
var item = combo.GetString(index)
```

参数*index*为索引。

返回值（Return Value）

字符串

示例 (Example)

```
var item4 = myCombo.GetString(4);
```

SetTopIndex

描述（Description）

设置列表框的第一个可见项；

语法（Syntax）

```
combo.SetTopIndex(index)
```

参数*index*索引的条目将被设为第一可见项。

返回值（Return Value）

无

示例 (Example)

```
myCombo.SetTopIndex(4);
```

GetTopIndex

描述（Description）
获取列表框的第一个可见项的索引值。

语法（Syntax）
```
var index= combo. GetTopIndex()
```

返回值（Return Value）
整型

示例 (Example)
```
var topIndex =myCombo.GetTopIndex();
```

SetEditMaxChars

描述（Description）
设置列表框项容纳最多的字符数。

语法（Syntax）
```
combo.SetEditMaxChars(max)
```
参数*max:*（整型）为文字输入框项可接受最多的字符数。

返回值（Return Value）
无

示例 (Example)
```
myCombo.SetEditMaxChars(10);
```

SetListWidth

描述（Description）

设置**ComboBox**列表的宽度。

语法（Syntax）

```
combo.SetListWidth(width)
```

参数*width:*（整型）为列表的宽度。

返回值（Return Value）

无

示例 (Example)

```
myCombo.SetListWidth(180);
```

ShowList

描述（Description）

显示或者隐藏列表框。

语法（Syntax）

```
combo.ShowList(show)
```

参数*show*（布尔值）： **true**指示列表框是否可见； **false**则隐藏列表框。

返回值（Return Value）

无

示例 (Example)

```
vmyCombo.ShowList(false);
```

SetCurSelection

描述（Description）
选择列表中的一项，并显示在文本框里。文本框里之前的内容会被清除。

语法（Syntax）
```
combo.SetCurSelection (index)
```
参数*index* 处的条目将被设置

返回值（Return Value）

示例 (Example)
```
myCombo. SetCurSelection (3)
```

ClearSelection

描述（Description）
清除文本框的内容，以及列表中被选择项的选中状态。

语法（Syntax）
```
combo. ClearSelection()
```

返回值（Return Value）
无

示例 (Example)
```
myCombo.ClearSelection()
```

GetCurrentSelection

描述（Description）

返回列表中当前被选中条目的索引值。

语法（Syntax）

```
var index = combo.GetCurrentSelection ()
```

返回值（Return Value）

无

示例 (Example)

```
<?xml version="1.0" encoding="utf-8" ?>
<x:elastos xmlns="http://www.elastos.com/XmlGlue/Widgets"
xmlns:x="http://www.elastos.com/XmlGlue/2.1">
	<window text="widget" left="0" top="0" width="240" height="320">
		<comboBox x:id="option" left="10" top="270" width="100" height="60">
				<item selected="true" text="test line" />
				<item text="Image 1" />
				<item text="Image 2" />
				<item text="Image 3" />
				<item text="Image 4" />
				<item text="Image 5" />
				<item text="pattern" />
		</comboBox>
		<pushButton x:id="button1" text="quit" left="180" top="250"
width="50" height="40" onSelect="Window.Close()" />
		<pushButton x:id="button2" text="Draw" left="120" top="250"
width="50" height="40" onSelect="OnSelect();" />
	</window>
	<x:script>
		<![CDATA[
		Grafix = Window.GetGrafix();
		img = new Image("romedalen.png");
		pat = Grafix.CreatePattern(img);
		function DrawLine() {
```

```
……
}
function DrawImage() {
……
}
function DrawImage2() {
……
}
function DrawImage3() {
……
}
function DrawImage4() {
……
}
function DrawImage5() {
……
}
function Draw6() {
……
}
function OnSelect()
{
selection = option.GetCurrentSelection ();
if (selection == 0) {
DrawLine();
}
else if (selection == 1) {
DrawImage();
}
else if (selection == 2) {
DrawImage2();
}
else if (selection == 3) {
```

```
                DrawImage3();
                }
                else if (selection == 4) {
                DrawImage4();
                }
                else if (selection == 5) {
                DrawImage5();
        }
        else if (selection == 6) {
                Draw6();
                }
                Grafix.Update();
        }
                ]]>
        </x:script>
</x:elastos>
```

SetFocus

描述（Description）

使控件获得焦点。

语法（Syntax）

```
        object.SetFocus () ;
```

返回值（Return Value）

无

注释（Remark）

示例 (Example)

```
        myCombo.SetFocus();
```

<listBox>标签 | ListBox 对象

命名空间URI （namespaceURI）

http://www.elastos.com/XmlGlue/Widgets

说明 （Description）

listBox 控件显示一个项列表，用户可从中选择一项或多项。通常，comboBox适合于存在一组“建议”选项的情况，而listBox适合于想要将输入限制为列表中内容的情况。comboBox包含一个文本框字段，因此可以键入列表中所没有的选项。

示例（Example）

```
        <window text="widget" left="10" top="10" width="200" height="200">
            <pushButton x:id="button1" text="quit" left="10" top="10" width="120"
height="40" onSelect="Window.Close();" />
            <listBox style="0x10000" text="list" left="10" top="60" width="180"
height="100">
                                <item>string1</item>
                                <item selected="true" text="string2" />
                                <item>string3</item>
        </listBox>
        </window>
```

JavaScript中

```
        var myListBox = new ListBox("list", ListBox::BORDER, 10, 60, 180, 100,);
        myListBox.AddString("string1");
        myListBox.AddString("string2");
        myListBox.AddString("string3");
        myListBox.SetSelection(1);
        function OnSelectionState(src, selectionState)
                {if (src == myListBox)
                print(selectionState);
        }
        myListBox.onSelectionState = OnSelectionState;
```

父元素 （Parent Element）

<window>

属性 （Attributes/Properties）

Attribute /Property	Value	Usage	Example	Description
onSelectionState		XML	`<listBox ... onSelectionState="print(selectionState)"> </listBox>`	当用户点击一个列表项目时，这个项目会被选中，产生 **onSelectionState** 事件。同时 **onSelectionState** 指定的事件处理程序或代码将被调用执行。 注：事件处理程序应包含一个参数：*selectionState*.其可能值为： ➢ BEFORECHANGE 用户在列表框的选择已经改变 ➢ CANCEL 用户取消了在列表框的选择
		JavaScript	`function OnSelelectionState(src, selectionState) { if (src == myListBox) print(selectionState); } myListBox.onSelectionState = OnSelectionState;`	

其他属性 见 通用属性

构造函数
ListBox

描述（Description）
创建一个列表项。

语法（Syntax）

```
var List = new ListBox (text, style, left, top, width, height)
```

text 为控件标题；该参数不可缺省。
style 为控件风格，可用的样式参见控件风格和ListBox控件风格。该参数不可缺省。
left 整数，指定按钮的左边到它的父亲容器的左边的距离，以屏幕像素为单位。该参数不可缺省。
top 整数，指定按钮的顶边到它的父亲容器的顶边的距离，以屏幕像素为单位。该参数不可缺省。
width 整数，指定按钮的水平尺寸，以屏幕像素为单位。该参数不可缺省。
height 整数，指定按钮的垂直尺寸，以屏幕像素为单位。该参数不可缺省。

注释（Remark）

示例 (Example)

```
var myList = new ListBox (“document1”,ListBox.MULTISELELECRION, 200,300,20,30)
```

方法

AddString
InsertString
DeleteString
GetItemCount
ResetContent
GetString
SetTopIndex
GetTopIndex
SetSelection
IsSelected
ClearSelection
GetSelectionCount
GetCurrentSelection
SetMultiSelection
SetFocus
其他方法 见 通用方法

AddString

描述（Description）
向列表框末尾添加一条项目。

语法（Syntax）

```
index=List.AddString(str)
```

参数*str*为所添加的项目。

返回值（Return Value）
返回集合中项的从零开始的索引。

示例 (Example)

```
var index = myList.AddString("Apple")';
```

InsertString

描述（Description）
向列表插入一条项目。

语法（Syntax）
```
index = List.InsertString(index, str)
```
参数*str*为所添加的项目。
参数*index*为列表项插入处的索引。

返回值（Return Value）
返回新添加条目的索引。

示例 (Example)
```
index=myList.InsertString(3,“Apple”)

myListBox.InsertString(0, "string0");
```

DeleteString

描述（Description）
删除列表框由参数*index*指示的条目。

语法（Syntax）
```
value = List. DeleteString(index)
```
参数*index*为将删除列表项的索引

返回值（Return Value）
返回剩余条目的总数。

示例 (Example)
```
var count = myList. DeleteString(3);
```

GetItemCount

描述（Description）
获取列表条目的总数。

语法（Syntax）

```
value = List. GetItemCount()
```

返回值（Return Value）
整型；列表框条目的总数。

示例 (Example)

```
var count= myList. GetItemCount()
```

ResetContent

描述（Description）
此方法用于清空控件的所有条目

语法（Syntax）

```
list.ResetContent()
```

返回值（Return Value）
无

示例 (Example)

```
myList.ResetContent();
```

GetString

描述（Description）
获取索引为参数*index*的条目值。

语法（Syntax）
```
item = list.GetString(index)
```

返回值（Return Value）
字符串

示例 (Example)
```
var item4 = myList.GetString(4)
```

SetTopIndex

描述（Description）
设置列表框的第一个可见项；参数*index*索引的条目将被设为第一可见项。

语法（Syntax）
```
list.SetTopIndex(index)
```

返回值（Return Value）
无

示例 (Example)
```
myList.SetTopIndex(4);
```

GetTopIndex

描述（Description）
获取列表框的第一个可见项的索引值。

语法（Syntax）

```
Index = list.GetTopIndex();
```

返回值（Return Value）
返回列表框的第一个可见项的索引值。

示例 (Example)

```
topindex = myList.GetTopIndex();
```

SetSelection

描述（Description）
设置条目为“选中”状态.

语法（Syntax）

```
list.SetSelection(index, bselected);
```

参数*index* 处的条目将被设置;
参数*bselected*为布尔值。

返回值（Return Value）
无

示例 (Example)

```
myList.SetSelection(3, true);
```

IsSelected

描述（Description）
检查返回某条目是否被选中。

语法（Syntax）

```
value = list. IsSelected (index);
```

参数*index* 为此条目的索引。

返回值（Return Value）
布尔值*:* true 表明被选中；false表明未被选中.

示例 (Example)

```
var b = myList. IsSelected (4);
```

ClearSelection

描述（Description）
重置选中状态。

语法（Syntax）

```
list. ClearSelection();
```

返回值（Return Value）
无

示例 (Example)

```
myList.ClearSelection();
```

GetSelectionCount

描述（Description）
获取被选中的条目的总数。

语法（Syntax）
```
value = list.GetSelectionCount();
```

返回值（Return Value）
返回被选中的条目的总数。

示例 (Example)
```
var SelectionCount = myList. GetSelectionCount ();
```

SetMultiSelection

描述（Description）
设置多个条目（连续或不连续）是否被选中。该方法只在列表框具有“多重选择”属性时可用。

语法（Syntax）
```
list.SetMultiSelection(first, last, selected);
```
参数*selected*为布尔值。
参数*first*和*last*分别表示起始和结束项的索引。

返回值（Return Value）

示例 (Example)
```
myList.SetMultiSelection(1,4,true);
```

GetCurrentSelection

描述（Description）
返回当前被选中条目的索引值。该方法只在列表框具有“单一选择”属性时可用。

语法（Syntax）

```
index = list.GetCurrentSelection ();
```

返回值（Return Value）
返回当前被选中条目的索引值。

示例 (Example)

```
SelectionIndex = myList. GetCurSelection();
```

SetFocus

描述（Description）
控件获得焦点。

语法（Syntax）

```
object.SetFocus () ;
```

返回值（Return Value）
无
注释（Remark）

示例 (Example)

```
myListbox.SetFocus();
```

<editBox>标签 | EditBox 对象

命名空间URI （namespaceURI）
http://www.elastos.com/XmlGlue/Widgets

说明 （Description）
编辑框 (**editBox**)控件用于显示,输入和修改数据。

示例（**Example**）

```
<editBox left="10" top="10" width="120" height="40">;
```

JavaScript中

```
var myEditBox = new EditBox("",0, 10, 10, 120, 40);
```

父元素 （**Parent Element**）

<window>

属性 （**Attributes/Properties**）

Attribute /Property	Value	Usage	Example	Description
readOnly	boolean	XML	<editBox left="10" top="10" width="120" height="40" readOnly="true"/>	此属性设置是否为只读。设置为只读后,文本框只能显示text属性中的内容,不能修改.
passwordChar	string	XML	<editBox eft="10" top="10" width="120" height="40" passwordChar="*"/>	设置用作占位符的字符。该字符用于输入屏蔽。
maxChars	int	XML	<editBox eft="10" top="10" width="120" height="40" maxChars="20"/>	设置编辑框最多可容纳的字符数。
onText		XML	<editBox left="10" top="10" width="120" height="40" onText="if (textState == xxx) do …;print(textState)"/>	当用户向编辑框中输入新信息，或当程序把**text**属性设置为新值从而改变其**text**属性时，将触发**onText**事件。此事件返回一个参数textState，其可能值为： ➢ UPDATE, 编辑框中的文字内容已经改变，即将输出到屏幕 ➢ CHANGE 编辑框中的文字内容已经改变，
		JavaScript	function onText(src, textState) { if (src == myEditBox) print(textState); } myEditBox.onText = OnSelectionState;	

				并且已经输出到屏幕 ➢ MAXTEXT 编辑框的文字内容已经不能在编辑框中完整显示（比如当用户输入已经超过了设定的最大长度时）

其他属性 见 通用属性

构造函数
EditBox

描述（Description）
创建一个按钮。

语法（Syntax）

```
EditBox TextBox = new EditBox (text, style, left, top, width, height);
```

text 为控件标题；该参数不可缺省。
style 为控件风格，可用的样式参见通用风格和EditBox风格。该参数不可缺省。
left 整数，指定控件的左边到它的父亲容器的左边的距离，以屏幕像素为单位。该参数不可缺省。
top 整数，指定控件的顶边到它的父亲容器的顶边的距离，以屏幕像素为单位。该参数不可缺省。
width 整数，指定控件的水平尺寸，以屏幕像素为单位。该参数不可缺省。
height 整数，指定控件的垂直尺寸，以屏幕像素为单位。该参数不可缺省。

注释（Remark）

示例 (Example)

```
myTextbox = new EditBox (“Warning”, EditBox::READONLY, 200,300,20,30);
```

方法
SetReadOnly
IsReadOnly
SetPasswordChar
SetMaxChars
ClearText
SetFocus
其他方法 见 通用方法

SetReadOnly

描述（Description）
设置编辑框的“只读”属性

语法（Syntax）

```
textbox.SetReadOnly(readOnly);
```

参数*readOnly*为布尔值：**true**，指示编辑框被设置为只读。

返回值（Return Value）

无

注释（Remark）

示例 (Example)

```
myTextbox.SetReadOnly(true);
```

IsReadOnly

描述（Description）

检查返回编辑框是否为“只读”。

语法（Syntax）

```
var value= textbox.IsReadOnly();
```

返回值（Return Value）

布尔值

注释（Remark）

示例 (Example)

```
var Value=myTextbox.IsReadOnly();
```

SetPasswordChar

描述（Description）
设置屏蔽密码字符串的字符（占位字符）。

语法（Syntax）
```
textbox.SetPasswordChar(str);
```
参数*str*为字符，此字符用于屏蔽单行编辑框中的密码字符。

返回值（Return Value）
无

注释（Remark）

示例 (Example)
```
myEditBox.SetPasswordChar("*");
```

SetMaxChars

描述（Description）
编辑框接受用户输入的最大字符数。

语法（Syntax）
```
textbox.SetMaxChars(max);
```
参数*max* 整型：指示最大字符数。

返回值（Return Value）
无

注释（Remark）

示例 (Example)
```
myTextbox.SetMaxChars(10);
```

ClearText

描述（Description）
清空编辑框的内容。

语法（Syntax）

```
textbox.ClearText();
```

返回值（Return Value）
无
注释（Remark）

示例 (Example)

```
myTextbox.ClearText();
```

SetFocus

描述（Description）
使控件获得焦点。

语法（Syntax）

```
object.SetFocus();
```

返回值（Return Value）
无
注释（Remark）

示例 (Example)

```
myTextbox.SetFocus();
```

<static>标签 | Static 对象

命名空间URI （namespaceURI）

http://www.elastos.com/XmlGlue/Widgets

说明 （Description）

本控件是静态文本框控件；可以显示文本字符串（这也是最常用的用法）和位图。

示例（Example）

```
        <static text="Image" left="10" top="110" width="120" height="108"
enableImage="true" image="test.png"/>;
```

JavaScript中

```
        var myStatic = new Static("Image", 0,10, 110, 120, 108);
        var img = new Bitmap("test.png");
        myStatic.image = img;
```

以上操作等价于：

```
        var myStatic = new Static("Image", 0,10, 110, 120, 108);
        myStatic.image = "test.png"
```

父元素 （Parent Element）

<window>

属性 （Attributes/Properties）

Attribute /Property	Value	Usage	Example	Description
enableImage	boolean	XML	`<static text="Image" left="10" top="110" width="120" height="108" enableImage="true"/>`	此属性设置静态文本框能否显示图片。
image		XML	`<static text="Image" left="10" top="110" width="120" height="108" enableImage="true" image="test.png"/>`	此属性设置静态文本框所显示图片文件的路径。
		JavaScript	`var img = new Bitmap("test.png");`	

			myStatic.image = img; myStatic.image = "test.png"	

其他属性 见 通用属性

构造函数

Static

描述（Description）

创建一个静态文本框控件。

语法（Syntax）

```
static = new Static (text, style, left, top, width, height);
```

text 为控件标题；该参数不可缺省。

style 为控件风格，可用的样式参见通用风格和Static控件风格。该参数不可缺省。

left 整数，指定控件的左边到它的父亲容器的左边的距离，以屏幕像素为单位。该参数不可缺省。

top 整数，指定控件的顶边到它的父亲容器的顶边的距离，以屏幕像素为单位。该参数不可缺省。

width 整数，指定控件的水平尺寸，以屏幕像素为单位。该参数不可缺省。

height 整数，指定控件的垂直尺寸，以屏幕像素为单位。该参数不可缺省。

注释（Remark）

示例 (Example)

```
var myStatic = new Static("Image", 0,10, 110, 120, 108);
```

方法

SetImage

GetImage

EnableImage

SetFocus

其他方法 见 通用方法

SetImage

描述（Description）

设置控件标签控件的图片.

语法（Syntax）

```
static.SetImage(img);
```

参数*img*为**Image**对象

返回值（Return Value）

无

注释（**Remark**）

示例 **(Example)**

```
var img = new Bitmap("test.png");
myStatic.SetImage(img);
```

GetImage

描述（**Description**）
获取**Static**控件的图形。

语法（**Syntax**）

```
imageObj = static.GetImage();
```

返回值（**Return Value**）
返回值为**Image**对象.

注释（**Remark**）

示例 **(Example)**

```
var myImage = myStatic.GetImage();
```

EnableImage

描述（**Description**）
设置控件是否接受图片。

语法（**Syntax**）

```
static.EnableImage(enabled);
```

参数*enabled*为布尔值

返回值（Return Value）

无

注释（Remark）

示例 (Example)

```
        myStatic.EnableImage(true);
```

SetFocus

描述（Description）

使控件获得焦点。

语法（Syntax）

```
        object.SetFocus ( ) ;
```

返回值（Return Value）

无

注释（Remark）

示例 (Example)

```
        myStatic.SetFocus();
```

<trackBar>标签 | TrackBar 对象

命名空间URI （namespaceURI）

http://www.elastos.com/XmlGlue/Widgets

说明 （Description）

trackBar表示一个标准的跟踪条。**trackBar**控件有两部分：游标和刻度线。

示例（**Example**）

```
        <trackBar style="0x01" left="200" top="10" width="20" height="200"
position="50"/>;
```

JavaScript中

```
        myTrackbar = new Trackbar("",Trackbar::VERT, 200, 10, 20, 200);
        myTrackbar.SetPosition(50);
```

父元素 （**Parent Element**）

<window>

属性 （**Attributes/Properties**）

Attribute /Property	Value	Usage	Example	Description
min	int	XML	`<trackBar style="0x01" left="200" top="10" width="20" height="200" position="50"/>`	设置此 trackBar使用的范围的下限.
max	int	XML	`<trackBar style="0x01" left="200" top="10" width="20" height="200" position="50"/>`	设置此 trackBar使用的范围的上限.
position	int	XML	`<trackBar style="0x01" left="200" top="10" width="20" height="200" position="50"/>`	设置表示游标在跟踪条上的当前位置的数值。
pageSize	int	XML	`<trackBar style="0x01" left="200" top="10" width="20" height="200" position="50" pageSize=”20”/>`	设置页步长。
onTrack		XML	`<trackBar style="0x01" left="200" top="10" width="20" height="200" position="50" onTrack="print(&quot;curPosition = &quot; + tracking);"/>`	当跟踪条的位置更改时，产生 **onTrack**事件。 注：事件返回一个参数:`tracking`. `tracking` 的可能值包括： TRUE：表示用户正在拖动游标； FALSE：表示用户停止拖动游标

		JavaScript	`myTrackbar = new Trackbar("",Trackbar::VERT 200, 10, 20, 200);`
			`myTrackbar.SetPosition(50);`
			`function OnTrack(src, tracking) {`
			`if (src == myTrackbar)`
			`print("curPosition = " + tracking);`
			`}`
			`myTrackbar.onTrack = OnTrack;`	

其他属性 见 通用属性

构造函数

TrackBar

描述（Description）

创建一个按钮。

语法（Syntax）

```
var trackbar = new TrackBar (text, style, left, top, width, height);
```

text 为控件标题；该参数可缺省。

style 为控件风格，可用的样式参见通用风格和TrackBar风格。该参数不可缺省。

left 整数，指定控件的左边到它的父亲容器的左边的距离，以屏幕像素为单位。该参数不可缺省。

top 整数，指定控件的顶边到它的父亲容器的顶边的距离，以屏幕像素为单位。该参数不可缺省。

width 整数，指定控件的水平尺寸，以屏幕像素为单位。该参数不可缺省。

height 整数，指定控件的垂直尺寸，以屏幕像素为单位。该参数不可缺省。

注释（Remark）

示例 (Example)

```
myTrackbar = new TrackBar (“”,TrackBar.AUTOTICKS, 200,300,20,30);
```

方法

SetRange
SetRangeMin
SetRangeMax
GetRangeMin
GetRangeMax
SetPosition
GetPosition
SetPageSize
GetPageSize
SetFocus

其他方法 见 通用方法

SetRange

描述（Description）

设置跟踪条的游标滑动范围。

语法（Syntax）

```
trackbar.SetRange(min, max);
```

参数*min*为最小值，参数*max*为最大值。

返回值（Return Value）

无

注释（Remark）

示例 (Example)

```
myTrackbar.SetRange(20,100);
```

SetRangeMin

描述（Description）

设置跟踪条的游标滑动的最小位置。

语法（Syntax）

```
trackbar.SetRangeMin(min);
```

参数*min*为整型，指示游标的最小位置。

返回值（Return Value）

无

注释（Remark）

示例 (Example)

```
myTrackbar.SetRangeMin(20);
```

SetRangeMax

描述（Description）

设置跟踪条的游标滑动的最大位置。

语法（Syntax）

```
trackbar.SetRangeMax(max);
```

参数*max*为整型，指示游标的最大位置。

返回值（Return Value）

无

注释（Remark）

示例 (Example)

```
        myTrackbar.SetRangeMax(200);
```

GetRangeMin

描述（Description）

获取跟踪条的游标滑动的起点(最小位置)。

语法（Syntax）

```
        Min=trackbar.GetRangeMin();
```

返回值（Return Value）

返回游标滑动的最小值。

注释（Remark）

示例 (Example)

```
        var min=myTrackbar.GetRangeMin();
```

GetRangeMax

描述（Description）

获取跟踪条的游标滑动的终点（最大位置）。

语法（Syntax）

```
Max=trackbar.GetRangeMax();
```

返回值（Return Value）
返回游标滑动的最大值。

注释（Remark）

示例 (Example)

```
var max=myTrackbar.GetRangeMax();
```

SetPosition

描述（Description）
设置跟踪条的游标当前位置。

语法（Syntax）

```
trackbar.SetPosition(position);
```

参数*position*指示当前位置的数值。

返回值（Return Value）
无

注释（Remark）

示例 (Example)

```
myTrackbar.SetPosition(80);
```

GetPosition

描述（Description）
获取跟踪条的游标当前位置。

语法（Syntax）
```
Position=trackbar.GetPosition();
```

返回值（Return Value）
返回游标当前位置的数值。

注释（Remark）

示例 (Example)
```
var position=myTrackbar.GetPosition();
```

SetPageSize

描述（Description）
设置游标滑动的页步长，页步长是指当用户在滑动轨道上单击时滑块所经过的长度。

语法（Syntax）
```
trackbar.SetPageSize(size);
```
参数*size*为整型，指示步长值。

返回值（Return Value）
无

注释（Remark）

示例 (Example)
```
myTrackbar.SetPageSize(20);
```

GetPageSize

描述（Description）
获取游标的步长。

语法（Syntax）

```
Pagesize = trackbar.GetPageSize();
```

返回值（Return Value）
返回游标的页步长

注释（Remark）

示例 (Example)

```
var pagesize=myTrackbar.GetPageSize();
```

SetFocus

描述（Description）
使控件获得焦点。

语法（Syntax）

```
object.SetFocus ( ) ;
```

返回值（Return Value）
无
注释（Remark）

示例 (Example)

```
myTrack.SetFocus();
```

<progressBar>标签 | ProgressBar 对象

命名空间URI （namespaceURI）

http://www.elastos.com/XmlGlue/Widgets

说明 （Description）

progressBar 控件通过在水平条中显示相应数目的矩形来指示操作的进度。操作完成时，进度栏被填满。进度栏通常用于帮助用户了解等待一项长时间的操作（例如，加载大文件）完成所需的时间。

示例（Example）

```
<progressBar left="10" top="240" width="200" height="20" position="50"/>;
```

JavaScript中

```
var myProgressBar = new ProgressBar("", 0, 10, 240, 200, 20);
myProgressBar.SetPosition(50);
```

父元素 （Parent Element）

<window>

属性 （Attributes/Properties）

Attribute /Property	Value	Usage	Example	Description
position	int	XML	`<progressBar left="10" top="240" width="200" height="20" position="50"/>`	本属性设置进度栏的当前位置；默认值为0。
step	int	XML	`<progressBar left="10" top="240" width="200" height="20" position="50" step="50"/>`	本属性设置进度栏推进的步长量。

其他属性 见 通用属性

构造函数

ProgessBar

描述（Description）

创建一个进度条控件。

语法（Syntax）

```
var pbar = new ProgessBar (text, style, left, top, width, height);
```

text 为控件标题；该参数可缺省。
style 为控件风格，可用的样式参见通用风格和Progress控件风格。该参数不可缺省。
left 整数，指定控件的左边到它的父亲容器的左边的距离，以屏幕像素为单位。该参数不可缺省。
top 整数，指定控件的顶边到它的父亲容器的顶边的距离，以屏幕像素为单位。该参数不可缺省。
width 整数，指定控件的水平尺寸，以屏幕像素为单位。该参数不可缺省。
height 整数，指定控件的垂直尺寸，以屏幕像素为单位。该参数不可缺省。

注释（Remark）

示例 (Example)

```
myPbar= new ProgessBar (“”, ProgressBar.SMOOTH, 200,300,20,30);
```

方法

SetRange
GetRangeMin
GetRangeMax
SetPosition
GetPosition
Increase
SetStep
Step
SetFocus
其他方法 见 通用方法

SetRange

描述（Description）

设置进度的范围。

语法（Syntax）

```
pbar.SetRange(min, max);
```

参数*min*为最小值，参数*max*为最大值。

返回值（Return Value）

无

注释（Remark）

示例 (Example)

```
myPbar.SetRange(20,100);
```

GetRangeMin

描述（Description）
获取进度的范围。

语法（Syntax）
```
Min=pbar.GetRangeMin();
```

返回值（Return Value）
返回进度条的最小值

注释（Remark）

示例 (Example)
```
var min=myPbar.GetRangeMin();
```

GetRangeMax

描述（Description）
获取进度条的最大值。

语法（Syntax）
```
Max=pbar.GetRangeMax();
```

返回值（Return Value）
返回进度条的最大值

注释（Remark）

示例 (Example)
```
var max=myPbar.GetRangeMax();
```

SetPosition

描述（Description）
设置进度的当前值.

语法（Syntax）

```
pbar.SetPosition(position);
```

参数*position*指示进度的当前值。

返回值（Return Value）
无
注释（Remark）

示例 (Example)

```
myPbar.SetPosition(80);
```

GetPosition

描述（Description）
获取进度的当前值。

语法（Syntax）

```
Position=pbar.GetPosition();
```

返回值（Return Value）
返回进度条的当前值。

注释（Remark）

示例 (Example)

```
var position=myPbar.GetPosition();
```

Increase

描述（Description）

按指定的数量增加进度栏的当前值.

语法（Syntax）

```
pbar.Increase(size);
```

参数*size*表示推进进度栏的当前值的增加量。

返回值（Return Value）

无

注释（Remark）

示例 (Example)

```
myPbar. Increase (5);
```

SetStep

描述（Description）

设置步长值.

语法（Syntax）

```
pbar.SetStep(int size);
```

返回值（Return Value）

无

注释（Remark）

示例 (Example)

```
myPbar. SetStep (5);
```

Step

描述（Description）
增加进度值一个步长。

语法（Syntax）

```
step = pbar.Step();
```

返回值（Return Value）
返回进度之前的值。

注释（Remark）

示例 (Example)

```
stepsize=myPbar.Step();
```

SetFocus

描述（Description）
使控件获得焦点。

语法（Syntax）

```
object.SetFocus ( ) ;
```

返回值（Return Value）
无
注释（Remark）

示例 (Example)

```
myPbar.SetFocus();
```

<item>标签

命名空间URI （namespaceURI）
http://www.elastos.com/XmlGlue/Widgets

说明 （Description）
该标签代表 <listBox>,<comboBox> 或<Menu> 元素中所包含一项.

示例（Example）

```

```

父元素 （Parent Element）
<listBox>
<comboBox>
<Menu>

属性 （Attributes/Properties）

Attribute /Property	Value	Usage	Example	Description
selected	Boolean	XML	<listBox style="0x10000" text="list" left="10" top="60" width="180" height="100"> <item>string1</item> <item selected="true" text="string2" /> <item>string3</item> </listBox>	设置该条项是否在其父 comboBox 或 listBox 中被选中. 值为**true,** 指示此项被父 comboBox 或 listBox中选中;否则为 **false.** **注意，此属性只对父元素为<comboBox>或 <listBox>有效.**
enabled	Boolean	XML	<menu> <item id="12" text="item1"/> <item id="13" enabled="false">item2</item> </menu>	设置该条项是否在其父 Menu 中被有效（即：对用户交互产生反应）： 为**true,** 指示此项有效；否则为 **false** **注意，此属性只对父元素为<menu>有效.**

Attribute /Property	Value	Usage	Example	Description
id	string	XML	<menu> <item id="17" checked="true">item5</item> </menu>	设置该菜单项的标示。**注意，此属性只对父元素为<menu>有效**
checked	Boolean	XML	<menu> <item id="17" checked="true">item5</item> </menu>	设置是否为该菜单项文字的左边做一个标记，即打一个勾：为**checked="true',** 指示此项被选中；否则**checked="false".** **注意，此属性只对父元素为<menu>有效**
text	string	XML	<menu> <item id="12" text="item1"/> …… </menu>	设置列表项或菜单项的内容。
			<comboBox x:id="option" left="10" top="270" width="100" height="60"> <item selected="true" text="test line"/> <item text="Image 1"/> <item text="Image 2"/> </comboBox>	
			<listBox style="0x10000" text="list" left="10" top="60" width="180" height="100"> <item>string1</item> <item selected="true" text="string2" /> <item>string3</item> </listBox>	

其他属性 见 公共属性

<separator>标签

命名空间URI （namespaceURI）
http://www.elastos.com/XmlGlue/Widgets

说明 （Description）
<separator>代表默认大小的分隔符，即：将一条直线添加到菜单的末尾。注意，本标签只对菜单元素(<menu>)有效。

示例（Example）

```
<menu onSelect="print(&quot;id = &quot; + id);">
        <item id="12" text="item1"/>
        <separator/>
        <item id="13" enabled="false">item2</item>
        <item text="item3">
                <item id="15" text="item4"/>
                <separator/>
                <item id="17" checked="true">item5</item>
                <separator/>
                <item id="16" text="item6"/>
        </item>
        <item id="18">item7</item>
        <separator/>
        <item id="19" text="item8"/>
</menu>
```

父元素 （Parent Element）
<menu>
<item>

属性 （Attributes/Properties）
无

Image 对象

说明 （Description）
代表一幅嵌入的图像。支持BMP JPEG PNG JIF EXIF ICON。

示例（Example）

下面的代码示例在JavaScript中创建了一个 Image对象， 并对它加载了图片.

```
var img = new Image();
img.LoadImage("XXX.png");
```

以上操作等价于：

```
var Img = new Image("XXX.png");
```

属性 （Properties）
无

构造函数
Image

描述（Description）
加载图片资源。

语法（Syntax）
```
var Img = new Image([imgfilepath]);
```
imgfilepath 字符串(可选)： 指示图片文件路径。

返回值（Return Value）
返回一个图片对象。

注释（Remark）

示例 (Example)

```
var Img = new Image(“dance4.png”);
```

方法小结
LoadImage
其他方法 见 通用方法

LoadImage

描述（Description）
加载图片资源。

语法（Syntax）

```
ImgObject.LoadImage(imgfilepath);
```

参数*imgfilepath* 字符串(可选)：指示图片文件路径。

返回值（Return Value）
无

注释（Remark）

示例 (Example)

```
var img = new Image();
img.LoadImage(“dancing6.png”);
```

Font 对象

说明 （Description）
指定用于渲染所包含文本的新字体、大小。

示例（Example）

下面的代码示例在JavaScript中创建了一个 **Font**对象.

```
var font = new Font("Arial", 20);
```

属性 （Properties）

Property	Value	Usage	Example	Description
weight	int	JavaScript	font.weight = 400;	本属性设置字体宽度（0-1000）。 如**weight**属性为400，字形为正常(normal)，则**weight** 为700时，字体为粗体； 如果**weigh**为0，则使用默认字形。
Size	int	JavaScript	font.size = 20;	本属性设置字体大小。

Property	Value	Usage	Example	Description
italic	boolean	JavaScript	`font.italic = true;`	本属性设置斜体。 **italic**属性为true，字体为斜体；
underline	boolean	JavaScript	`font.underline = false;`	本属性设置字体下划线。 **underline**属性为true，字体显示下划线； 否则，**underline**属性为false.
strikeout	boolean	JavaScript	`font.strikeout = false;`	本属性设置字体是否显示删除线。 **strikeout**属性为true，字体显示删除线； 否则，**strikeout**属性为false.

构造函数

Font

描述（Description）

生成一个Font对象。

语法（Syntax）

```
var font = new Font(faceName, size);
```

faceName（必须）字符串（不超过32个字节）：为当前的字样家族.

size（必须）整型*:*该对象的字体大小。

返回值（Return Value）

返回一个字体对象。

注释（Remark）

示例 (Example)

```
var font = new Font("Arial", 20);
```

方法小结

无

Pattern 对象

说明 （Description）

XGrafix对象的填充模式，包括：图形填充，渐变填充（包括：径向渐变填充，或线性渐变填充等）等。

示例（Example）

下面的代码示例在JavaScript中创建了一个Pattern对象.

```
var pattern = GrafixObject.CreatePattern(155, 155, 155);
```

属性 （Properties）

Attribute /Property	Value	Usage	Example	Description
extend	“none”:透明； **“**repeat”：图像按平铺模式重复； “reflect”：反射 “pad”：复制距离模版最近的像素	JavaScript	`pattern.extend ="repeat";`	本属性设置模版以外区域的填充模式。
filter	“fast” “good” “best” “nearest” “bilinear” “gaussian”	JavaScript	`pattern.filter = "good";`	本属性设置过滤。

构造函数

CreatePattern
CreateLinearGradient
CreateRadialGradient

方法小结

AddColorStop
Scale
Translate
Rotate

AddColorStop

描述（Description）

设置终止颜色。

语法（Syntax）

```
pattern.AddColorStop(offset, r, g, b, a);
```

或

```
pattern.AddColorStop(offset, r, g, b);
```

r g b 分别指明三原色中红色、绿色、蓝色的强度，它们的取值范围为0～255
*a*用于指定颜色的透明度。取值范围为0～1：0 表示完全透明，1 表示完全不透明。
参数*offset*：颜色的起始索引（沿渐变控制条的位移量），取值范围为0～1

返回值（Return Value）
无

注释（Remark）
如果**Pattern**是线性（liner）或辐射形(radial)模式，而不是渐变模式，**Pattern**的状态被置成错误状态--pattern type mismatch。

示例 (Example)

```
var pattern = GrafixObject.CreateLinearGradient(25, 25, 125, 125);
pattern.AddColorStop( 0, 0, 0, 0, 0);
pattern.AddColorStop( 1, 255, 0, 0, 1);
```

Scale

描述（Description）
本方法为缩放变换。

语法（Syntax）

```
pattern.Scale(sx, sy);
```

参数*sx*：（必须）整形， 为x坐标的放缩因子;
参数*sy*：（必须）整形，为y坐标的放缩因子.

将每一点的横坐标放大（缩小）至*sx*倍，纵坐标放大（缩小）至*sy*倍. 即：每一点移动到（x * *sx*， y **sy*）

返回值（Return Value）
无

注释（Remark）

示例 (Example)

```
var img = new Image("XXX.png");
var pattern = GrafixObject.CreatePattern(img);
pattern.Scale(2, 3);
```

Translate

描述（Description）
本方法为平移变换。

语法（Syntax）

```
pattern.Translate(tx, ty);
```

参数*tx*：（必须）整形， 为x坐标的平移量;
参数*ty*： （必须）整形，为y坐标的平移量.
即：将每一点移动到(x+tx, y+ty)。

返回值（Return Value）
无

注释（Remark）

示例 (Example)

```
var img = new Image("XXX.png");
var pattern = GrafixObject.CreatePattern(img);
pattern.Translate(10, 10);
```

Rotate

描述（Description）
方法将图形旋转一个角度。

语法（Syntax）

```
pattern.Rotate(angleInRadians);
```

参数*angleInRadians* 为旋转弧度值。

注意：*angleInRadians* = 旋转角度值（q） * Math.PI */180*
即： 将每一点(x,y) 移动到（x*Cosq – y*Sinq，x*Sinq + y*Cosq）

返回值（Return Value）
无

注释（Remark）

示例 (Example)

```
var img = new Image("XXX.png");
var pattern = GrafixObject.CreatePattern(img);
var rad = 45 * Math.PI / 180; // 45 degrees
pattern.Rotate(rad);
```

XGrafix 对象

说明 （Description）
XGrafix对象是画图时需要使用的对象。使用**XGrafix**对象进行绘图，通常先设置绘图选项（**lineWidth**、**pattern**），用函数（例如：**MoveTo， LineTo**）创造图样，再用函数（例如 **Stroke， Fill**）绘制图样。

示例（Example）

下面的代码示例在JavaScript中创建了一个**XGrafix**对象.

```
var GrafixObject = new XGrafix(240, 300);
```

属性 （Properties）

Property	Value	Usage	Example	Description
lineWidth	int	JavaScript	`GrafixObject.lineWidth = 2;`	本属性设置或获取当前XGrafix对象的直线宽度（即：画笔的直径）。

Property	Value	Usage	Example	Description
				默认值为2.0.
miterLimit	int	JavaScript	`GrafixObject.miterLimit = 10;`	本属性设置当前**XGrafix**对象的斜接面限量。 注意，当lineJoin = Miter时有效,用来判断是否用bevel来代替Miter(即：当斜接面除以lineWidth>miterLimit时, lineJoin =bevel。
font	JSObject	JavaScript	`var font = new Font("Arial", 20);` `GrafixObject.font = font;`	本属性设置或获取当前XGrafix对象的字体。

pattern	JSObject	JavaScript	`var img = new Image("XXX.png");`
			`var pattern = GrafixObject.CreatePattern(img);`
			`GrafixObject.pattern = pattern;`
				本属性设置或获取当前XGrafix对象的填充模式。
				本填充模式将在当前XGrafix对象一直被使用，直到指定新的模式。

lineCap	“butt”: 平线帽 “round”：圆线帽“ square”: 方线帽	JavaScript	`GrafixObject.lineCap = "round";`	本属性设置或获取当前XGrafix对象的线帽样式(线帽用于结束绘制的直线).
lineJoin	“Miter”：斜联接； “round”**:** 光滑连接，在两条线之间产生平滑的圆弧 “bevel”：斜切（指定成斜角的联接。这将产生一个斜角）		`GrafixObject.lineJoin = "miter";`	本属性设置或获取当前XGrafix对象的笔划连接(笔划连接.定义了绘制折线的角时使用的形状)
fillRule	“winding”: 缠绕填充算法 “even-odd”：奇偶（交错）填充算			本属性设置或获取当前XGrafix对象的填充规则（填充规则对所有路径区域填充或者只是对交叉部分填充）。 当前填充规则影响**Fill** 和**Clip** 方法）

Property	Value	Usage	Example	Description
	法。			
compositeOperation	**clear** **source** **over** **inout** **atop** **dest** **dest-over** **dest-int** **dest-out** **dest-atop** **xor** **add** **saturate**		`GrafixObject.compositeOperation = "over";`	本属性设置或获取当前XGrafix对象的组合操作的模式。

构造函数
XGrafix

描述（Description）
用于创建一个**XGrafix**对象.

语法（Syntax）

```
var GrafixObject = new XGrafix(width, height);
```

width：（整型）指明对象宽度；
height：（整型）指明对象高度。

返回值（Return Value）
返回一个**XGrafix**对象

注释（Remark）

示例 (Example)

```
var GrafixObject = new XGrafix(240, 300);

var GrafixObject = Window.GetGrafix();
```

方法小结

BeginPath
Arc
BezierCurveTo
MoveTo
LineTo

Rect
ClosePath
CreateLinearGradient
CreatePattern
CreateRadialGradient
Fill
Stroke
Clip
Paint
DrawImage
FillRect
StrokeRect
ClearRect
Scale
Translate
Rotate
Save
Restore
ShowText
SetDash
SetPattern
Update

方法
BeginPath

描述（Description）
调用启动一个路径。在这个命令后执行的绘图方法会自动成为路径的一部分。

语法（Syntax）

```
        GrafixObject.BeginPath();
```

返回值（Return Value）
空

注释（Remark）

要使用路径绘制图形，需要一系列特殊的步骤：
BeginPath()
ClosePath()
Stroke()
Fill()
第1步调用**BeginPath**方法创建路径，在系统内部，路径是作为一系列子路径（线、弧等）列表存储的，他们一起来构造成图形，每次这个方法调用的时候，该列表将被轻空重置，然后我们就可以开始制新的图形了。
第2步是做实际的绘制工作，使用绘制方法（如：**Arc 或 Rect**方法）。

第3个可选的步骤，调用**ClosePath**方法，该方法试图在起始点和当前点之间绘制线条来封闭图形，如果该图形实际已经封闭或者列表中只有一个点，该函数什么效果都没有。
最后的步骤将要调用笔触(**Stroke**方法)或/和填充（**Fill**方法），调用其中之一将实际在面板上绘制出图形。 笔触用来图形轮廓而填充方法用来填充块状区域图形。

示例 (Example)

```
GrafixObject.BeginPath();
GrafixObject.Rect( 0, 0, 100, 120);
GrafixObject.Stroke();
```

Arc

描述（Description）
画弧形或者圆形。

语法（Syntax）

```
GrafixObject.Arc(x, y, radius, startAngle, endAngle, anticlockwise);
```

或

```
GrafixObject.Arc(x, y, radius, startAngle, endAngle);
```

参数*x*和*y*表示圆弧的中心坐标为（*x, y*）；
参数*radius*是圆的半径；
参数*startAngle*和*endAngle*是圆弧起始和截至点弧度，起始和截至点弧度是基于x坐标测量的；
参数*anticlockwise* 布尔型(是否逆时针)：true表示逆时针方向绘制圆弧，false则为顺时针方向。

注意，Arc方法中的角度是弧度而不是度数，您可以使用该Javascript代码转换角度到弧度：
var radius = (Math.PI/180)*degrees。

返回值（Return Value）
无

注释（Remark）

示例 (Example)

```
        GrafixObject.BeginPath();
        GrafixObject.Arc(75, 75, Math.PI / 2, 30, 0, true); // or GrafixObject.arc(75,
75, Math.PI /2,30, 0);
        GrafixObject.Stroke();
```

BezierCurveTo

描述（Description）
绘制复杂的自然曲线(即：贝塞尔曲线)。贝塞儿曲线有两个控制点。

语法（Syntax）

```
        GrafixObject.BezierCurveTo(cp1x, cp1y, cp2x, cp2y, x, y );
```

参数*x, y*指的都是截至点的坐标；
参数*cp1x, cp1y*是第1个控制点的坐标；
参数*cp2x, cp2y*是第2个控制点坐标。

返回值（Return Value）
无

注释（Remark）

示例 (Example)

```
        GrafixObject.BeginPath();
        GrafixObject.MoveTo(0, 0);
        GrafixObject.BezierCurveTo(0, 150, 150, 0, 150, 150);
        GrafixObject.Stroke();;
```

MoveTo

描述（Description）
本方法并不实际绘制任何东西，但是这是一个非常的重要的方法，它是路径列表的一部分。可以这样来理解该方法的作用，就如你的钢笔或者铅笔在纸上的一点移动到下一点。

调用本方法将封闭当前的子路径，创建一个新的子路径。

语法（Syntax）

```
GrafixObject.MoveTo(x, y);
```

参数*x, y* 分别表示将要移动到的新的位置点的x,y轴坐标。

返回值（Return Value）

无

注释（Remark）

示例 (Example)

```
GrafixObject.BeginPath();
GrafixObject.MoveTo(10, 10);
GrafixObject.LineTo(75, 65);
GrafixObject.LineTo(140, 10);
GrafixObject.MoveTo(10, 140);
GrafixObject.LineTo(75, 80);
GrafixObject.LineTo(140, 140);
GrafixObject.Fill();
```

LineTo

描述（Description）

本方法用于绘制直线.

语法（Syntax）

```
GrafixObject.LineTo(x, y);
```

参数*x, y* 标明要绘制线条的截至点（终点）

返回值（Return Value）

无

注释（Remark）
直线的起始点与前面的绘制路径有关，前次路径的截至点就是本次路径的起始点，起始点可以通过**MoveTo**方法改变。

示例 (Example)

```
GrafixObject.BeginPath();
GrafixObject.MoveTo( 10, 140 );
GrafixObject.LineTo( 75, 10 );
GrafixObject.LineTo( 140, 140 );
GrafixObject.Stroke();
```

Rect

描述（Description）
在路径上添加矩形。

语法（Syntax）

```
GrafixObject.Rect(x, y, width, height);
```

参数 *x, y* 是矩形路径的左上角坐标；
参数 *width, height* 定义了矩形的宽和高。

返回值（Return Value）
无

注释（Remark）
执行完这个方法后，**MoveTo**方法将被自动执行(重置起始绘制点到默认位置(x, *y*)）.

示例 (Example)

```
GrafixObject.BeginPath();
GrafixObject.lineWidth = 4;
GrafixObject.Rect( 10, 10, 75, 75 );
GrafixObject.Stroke();
```

ClosePath

描述（Description）
在路径的起始点和当前点之间绘制线条来封闭图形。

语法（Syntax）

```
GrafixObject.ClosePath();
```

返回值（Return Value）
无

注释（Remark）

Fill() 或 **Clip()** 方法将自动封闭开放路径。
Stroke（）方法不封闭路径。

示例 (Example)

```
GrafixObject.ClosePath();
```

CreateLinearGradient

描述（Description）
创建一个线形渐变的模式。

语法（Syntax）

```
GrafixObject.CreateLinearGradient(x0, y0, x1, y1);
```

按照从点1（*x0, y0*）到点2（*x1, y1*） 这种路径进行梯度渐变。

返回值（Return Value）
无

注释（Remark）

示例 (Example)

```
var pattern = GrafixObject.CreateLinearGradient(25, 25, 125, 125);
pattern.AddColorStop( 0, 0, 0, 0, 0);
pattern.AddColorStop( 1, 255, 0, 0, 1);
GrafixObject.pattern = pattern;
GrafixObject.FillRect(25, 25, 125, 125);
```

CreatePattern

描述（Description）
创建一个模式。

语法（Syntax）

```
GrafixObject.CreatePattern(Grafix);
GrafixObject.CreatePattern(img);
GrafixObject.CreatePattern(r, g, b);
GrafixObject.CreatePattern(r, g, b, a);
```

参数*Grafix* 为一个**XGrafix**对象。注意，**XGrafix**对象应通过 “new XGrafix（*width，height*）”获得；
参数*img* 为一个**Image**对象；
参数*r,g, b* 分别指明三原色中红色、绿色、蓝色的强度，它们的取值范围为0～255；
参数*a* 用于指定颜色的透明度。取值范围为0～1：*0* 表示完全透明，1 表示完全不透明。

返回值（Return Value）
无

注释（Remark）

示例 (Example)

示例1

```
var Grafix = new XGrafix(240, 320);
var pattern = GrafixObject.CreatePattern(Grafix);
var pattern = GrafixObject.CreatePattern(125, 125, 125);
Grafix.pattern = pattern;
```

示例2

```
var Grafix = new XGrafix(240, 320);
var pattern = GrafixObject.CreatePattern(Grafix);
var pattern = GrafixObject.CreatePattern(125, 125, 125, 0.5);
Grafix.pattern = pattern;
```

示例3

```
var Grafix = new XGrafix(240, 320);
var img = new Image("XXX.png");
var pattern = GrafixObject.CreatePattern(img);
Grafix.pattern = pattern;
```

CreateRadialGradient

描述（Description）

创建一个径向渐变的模式。

语法（Syntax）

```
GrafixObject.CreateRadialGradient(x0, y0, r0, x1, y1, r1);
```

参数（Parameters）

x0 int 起始圆心的 x 坐标
y0 int 起始圆心的 y 坐标
r0 int 起始圆心的半径
x1 int 外圈圆心的 x 坐标
y1 int 外圈圆心的 y 坐标
r1 int 外圈圆心的半径

返回值（Return Value）
无

注释（Remark）

示例 (Example)

```
Var pattern = GrafixObject.CreateRadialGradient( 75, 75, 10, 70, 70, 40 );
pattern.AddColorStop( 0, 255, 0, 0 );
pattern.AddColorStop( 0.7, 102, 0, 0);
pattern.AddColorStop( 1, 33, 0, 0, 0);
GrafixObject.pattern = pattern;
GrafixObject.FillRect( 0, 0, 150, 150 );
```

Fill

描述（Description）
通过从当前点到子路径的起始点之间画一条直线，而封闭开放路径。通常，**Fill**(填充)方法用来填充块状区域图形。

语法（Syntax）

```
GrafixObject.Fill([bPreserve]);
```

参数*bPreserve:*布尔型（可选）

返回值（Return Value）
无

注释（Remark）
如果 *bPreserve*= true，则函数执行后，不擦除当前路径。

```
GrafixObject.Rect(10, 10, 200, 120);
GrafixObject.Fill(true);
GrafixObject.Stroke();
```

以上一组操作等价于下面的一组操作：

```
GrafixObject.Rect(10, 10, 200, 120);
GrafixObject.Fill();
GrafixObject.Rect(10, 10, 200, 120);
GrafixObject.Stroke();
```

示例1 (Example)

```
GrafixObject.Rect(10, 10, 200, 120);
GrafixObject.Fill(true);
GrafixObject.Stroke();
```

示例2 (Example)

```
GrafixObject.Rect(10, 10, 200, 120);
GrafixObject.Fill();
GrafixObject.Rect(10, 10, 200, 120);
GrafixObject.Stroke();
```

Stroke

描述（Description）

为图形勾轮廓。本方法并不封闭路径。

语法（Syntax）

```
GrafixObject.Stroke([bPreserve]);
```

参数*bPreserve:*布尔型（可选）

返回值（Return Value）

无

注释（Remark）

如果 *bPreserve*= true，则函数执行后，不擦除当前路径。

```
GrafixObject.Rect(10, 10, 200, 120);
```

```
GrafixObject.Stroke(true);
GrafixObject.Fill();
```

以上一组操作等价于下面的一组操作：

```
GrafixObject.Rect(10, 10, 200, 120);
GrafixObject.Stroke();
GrafixObject.Rect(10, 10, 200, 120);
GrafixObject.Fill();
```

示例1 (Example)

```
GrafixObject.Rect(10, 10, 200, 120);
GrafixObject.Stroke(true);
GrafixObject.Fill();
```

示例2 (Example)

```
GrafixObject.Rect(10, 10, 200, 120);
GrafixObject.Stroke();
GrafixObject.Rect(10, 10, 200, 120);
GrafixObject.Fill();
```

Clip

描述（Description）

用当前路径剪切画布。

语法（Syntax）

```
GrafixObject.Clip([bPreserve]);
```

参数*bPreserve:*布尔型（可选）

返回值（Return Value）

无

注释（Remark）

同 **Fill** 和 **Stroke** 方法，如果参数 *bPreserve*= true，则函数执行后，不擦除当前路径。

示例 (Example)

```
GrafixObject.Rect(10, 10, 200, 120);
GrafixObject.Clip();
GrafixObject.Fill();
```

Paint

描述（Description）

使用当前的模式(Pattern),去填充当前的剪切图形。

语法（Syntax）

```
GrafixObject.Paint([alpha]);
```

参数*alpha:*整型（可选），用于指定颜色的透明度。取值范围为0～1：*0* 表示完全透明，1 表示完全不透明。

返回值（Return Value）

无

注释（Remark）

示例 (Example)

```
var img = New Image("XXX.png");
GrafixObject.pattern = GrafixObject.CreatePattern(img);
GrafixObject.Rect(10, 10, 200, 120);
GrafixObject.Clip();
GrafixObject.Paint(0.5); // or GrafixObject.Paint();
```

DrawImage

描述（Description）
本方法用于绘图。

语法（Syntax）

```
GrafixObject.DrawImage(image, x, y );// 第一种方式
GrafixObject.DrawImage(image, x, y, width, height );// 第二种方式
GrafixObject.DrawImage(image, srcX, srcY, srcWidth, srcHeight,
dstX, dstY, dstWidth, dstHeight);// 第三种方式
```

srcX：要绘制的源图像部分的左上角的 x 坐标。
srcY：要绘制的源图像部分的左上角的 y 坐标。
srcWidth：要绘制的源图像部分的宽度。
srcHeight：要绘制的源图像部分的高度。
dstX：要绘制的目标图像部分的左上角的 x 坐标
dstY：要绘制的目标图像部分的左上角的 y 坐标
dstWidth：要绘制的目标图像部分的宽度
dstHeight：要绘制的目标图像部分的高度

第一种方式：按照图片（*image*）的原始尺寸，将其绘制在指定位置（*x, y*），这个位置和*image*图像的左上角点相对应。
第二种方式：按照宽（*width*）、高（*height*）将图形（*image*）绘制在指定位置（*x, y*），这个位置和*image*图像的左上角点相对应。
第三种方式：将位于（*srcX, srcY*），按大小为（*srcWidth, srcHeight*）的源图形（*image*）绘制在指定位置（*dstX, dstY*），目标图形的高度为*dstHeight*　宽度为*dstWidth*。

返回值（Return Value）
无

注释（Remark）

示例 (Example)

```
var img = new Image("me.png");
GrafixObject.DrawImage(img, 0, 0, 240, 300);
```

FillRect

描述（Description）
用当前模式填充一个矩形。

语法（Syntax）

```
GrafixObject.FillRect(x, y, width, height);
```

（*x, y*）表示矩形左上角在面板的位置（相对原点）
width （整型）：要绘制的目标图像部分的宽度
height（整型）：要绘制的目标图像部分的高度

返回值（Return Value）
无

注释（Remark）

示例 (Example)

```
GrafixObject.FillRect(10, 10, 120, 120);
```

StrokeRect

描述（Description）
绘制矩形框。

语法（Syntax）

```
GrafixObject.StrokeRect(x, y, width, height);
```

（*x, y*）表示矩形左上角在面板的位置（相对原点）
width （整型）：要绘制的目标图像部分的宽度
height（整型）： 要绘制的目标图像部分的高度

返回值（Return Value）
无

注释（Remark）

示例 (Example)

```
        GrafixObject.StrokeRect(10, 10, 120, 120);
```

ClearRect

描述（Description）
清除特定的矩形区域并且使该区域完成透明。

语法（Syntax）

```
        GrafixObject.ClearRect(x, y, width, height);
```

（*x, y*）表示矩形左上角在面板的位置（相对原点）
width （整型）：要清除的目标图像的宽度
height（整型）： 要清除的目标图像的高度

返回值（Return Value）
无

注释（Remark）

示例 (Example)

```
        GrafixObject.ClearRect(10, 10, 120, 120);
```

Scale

描述（Description）
缩放变换.

语法（Syntax）

```
XGrafixObject.Scale(sx, sy)
```

参数*sx*：（必须）整形， 为x坐标的放缩因子;
参数*sy*：（必须）整形，为y坐标的放缩因子.
将每一点的横坐标放大（缩小）至*sx*倍，纵坐标放大（缩小）至*sy*倍. 即：每一点移动到（x * *sx*， y *sy）

返回值（Return Value）
无

注释（Remark）
如需取消所设置的放缩变换操作，并将选定的对象还原到未变换前的样式，请使用**Save**和 **Restore**方法，适当保存或者恢复图形对象的状态。

示例 (Example)

```
XGrafixObject.Scale(5, 5);
```

Translate
描述（Description）
平移变换。

语法（Syntax）

```
XGrafixObject.Translate( dx, dy );
```

参数*dx*：（必须）整形， 为x坐标的平移量;
参数*dy*：（必须）整形，为y坐标的平移量.
即：将每一点移动到(x+dx, y+dy)。

返回值（Return Value）
无

注释（Remark）
如需取消所设置的平移变换操作，并将选定的对象还原到未变换前的样式，请使用**Save**和 **Restore**方法，适当保存或者恢复图形对象的状态。

示例 (Example)

```
XGrafixObject.Translate( 75, 75 );
```

Rotate

描述（Description）
将图形旋转一个角度。

语法（Syntax）

```
XGrafixObject.Rotate(angleInRadians );
```

参数*angleInRadians* 为旋转弧度值
注：*angleInRadians* = 旋转角度值（q） * Math.PI */180*
即 将每一点(x,y) 移动到（x*Cosq – y*Sinq，x*Sinq + y*Cosq）

返回值（Return Value）
无

注释（Remark）
如需取消所设置的旋转变换操作，并将选定的对象还原到未变换前的样式，请使用**Save**和 **Restore**方法，适当保存或者恢复图形对象的状态。

示例 (Example)

```
rad = 45 * .01745329252; // 角度 45 的弧度值
XGrafixObject.Save();
XGrafixObject.Translate( 75, 75 );
XGrafixObject.Rotate( rad );
XGrafixObject.StrokeRect( -50, -50, 100, 100 );
XGrafixObject.Restore();
```

Save

描述（Description）
将保存XGrafix的状态信息块在堆栈上。

语法（Syntax）

```
GrafixObject.Save();
```

返回值（Return Value）
无

注释（Remark）

Save 方法保存的**XGrafix**的状态信息块包括以下：

- **translation matrix (CTM)**
- **clip**
- **lineCap**
- **lineWidth**
- **lineJoin**
- **miter limit**
- **pattern**

注意，**Save**方法不保存当前路径。
注意，**Save**方法与**Restore**方法应成对使用。

示例 (Example)

```
GrafixObject.Save();
```

Restore
描述（Description）
还原到状态堆栈中最近保存的状态。

语法（Syntax）

```
GrafixObject.Restore();
```

返回值（Return Value）
无

注释（Remark）

调用**Restore** 方法时，信息块被从堆栈中移除并且用于将 **XGrafix** 对象还原到它在 **Save** 方法调用时所处的状态。

本方法与**Save**方法成对使用。详细信息，请参见**Save**方法。

示例 (Example)

```
rad = 45 * .01745329252; // 45 degrees
XGrafixObject.Save();
XGrafixObject.Translate( 75, 75 );
XGrafixObject.Rotate( rad );
XGrafixObject.StrokeRect( -50, -50, 100, 100 );
XGrafixObject.Restore();
```

ShowText

描述（Description）

在指定坐标显示字符串。

语法（Syntax）

```
GrafixObject.ShowText(x, y, text);
```

（*x*，*y*）为显示文字的起始位置；

text 为所需显示的字符串。

返回值（Return Value）

无

注释（Remark）

示例 (Example)

```
GrafixObject.ShowText(10, 10, "hello world!");
```

SetDash

描述（Description）
设置虚线的模式，此处设置的虚线的模式将被**Stroke**方法使用。

语法（Syntax）
```
GrafixObject.SetDash(array, offset);
```
参数*array*：虚线样式数组。数组内的值为空白的交错值。
注意，本数组应由正数组成。如果数组无成员，虚线被禁止；如果数组只有1个成员，对称模式将被使用，即："on"段 和 "off" 段的长度均等于该数值。

参数*offset*：虚线样式数组的起始索引。

返回值（Return Value）
无

注释（Remark）
如果*array*数组中有负值，或*array*数组中全为0，将报错“invalid dash”

示例 (Example)
```
dashes = new Array(3);
dashes[0] = 10;
dashes[1] = 20;
dashes[2] = 10;
GrafixObject.SetDash(dashes, 5);
```

SetPattern

描述（Description）
设置填充模式。

语法（Syntax）

```
GrafixObject.SetPattern(pattern);
GrafixObject.SetPattern(r, g, b, a);
GrafixObject.SetPattern(r, g, b);
```

pattern ：一个现有的Pattern对象。
r g b 分别指明三原色中红色、绿色、蓝色的比例，它们的取值范围为0～255
a 指定颜色的透明度：0 表示完全透明，1 表示完全不透明, 取值范围为0～1

返回值（Return Value）
无

注释（Remark）
你设定的模式会一直生效，直到设定新的模式。

示例1 (Example)

```
GrafixObject.SetPattern(125, 125, 125);
GrafixObject.FillRect(10, 10 ,120, 120);
```

示例2 (Example)

```
var pattern = GrafixObject.CreatePattern(125, 125, 125);
GrafixObject.SetPattern(pattern);
GrafixObject.FillRect(10, 10 ,120, 120);
```

Update

描述（Description）
刷新当前显示区域。

语法（Syntax）

```
GrafixObject.Update();
```

返回值（Return Value）

无

注释（Remark）

本方法只对通过 Window.GetGrafix() 获得的对象有效。

示例 (Example)

```
GrafixObject.FillRect(10, 10 ,120, 120);
GrafixObject.Update();
```

Application 对象

说明 （Description）

Application是一个全局对象，可以通过名字（Application）直接访问。用来加载新的window或者结束应用程序。

注意，可以通过名字直接访问Application 对象，无需创建对象.

示例（Example）

下面的代码示例在JavaScript中，使用Application对象加载一个窗口。

```
var newWnd = Application.LoadWindow("main.xml");
newWnd.Show();
```

属性 （Properties）

无

构造函数

无

方法小结

LoadWindow

Quit

方法

LoadWindow

描述（Description）

根据用户指定的通用资源标识符加载一个新的窗口（**window**）。

语法（Syntax）

```
Application.LoadWindow(uri);
```

参数*uri：* 窗口的通用资源标识符(URI)

返回值（Return Value）
window对象

注释（Remark）

示例 (Example)

```
        var newWnd = Application.LoadWindow("test.xml?arg1=value1&arg2=value2");
        newWnd.Show();
```

Quit

描述（Description）
结束当前应用。

语法（Syntax）

```
        Application.Quit();
```

返回值（Return Value）
无

注释（Remark）

示例 (Example)

```
<x:elastos xmlns="http://www.elastos.com/XmlGlue/Widgets"
xmlns:x="http://www.elastos.com/XmlGlue/2.1">
        <window text="widget" left="0" top="10" width="240" height="280"
onTimer="OnTimer(timerId)">
            ……
        </window>
        <x:script>
        <![CDATA[
```

```
            function OnSelect1()
            {
            var newWnd = Application.LoadWindow("main.xml");
            newWnd.Show();
            }
            function OnSelect2()
            {
            Application.Quit();
            }
            button1 = new PushButton("NewWnd", 0, 130 ,200 ,100, 40);
            button1.onSelect = OnSelect1;
            button2 = new PushButton("Quit", 0, 10 ,200 ,100, 40);
            button2.onSelect = OnSelect2;
            ]]>
            </x:script>
</x:elastos>
```

1.4 全局函数

本版本包括全局函数：
print
loadModule

Print

描述（Description）
将字符串以后接一个换行符，并发送到控制台（如：屏幕）。常用于调试。

语法 （Syntax）

```
        print(str);
```

参数 （Parameters）

Parameter	Description
str	将被输出的字符串

返回值 （Return Values）
无

示例 （**Example**）

```
print("idx = " + idx);
```

loadModule

描述（**Description**）

加载用户的动态连接库（DLL）;Dll必须以c函数方式导出GetModuleObject方法。GetModuleObject的原型为JSObject * (* GetModuleObject)(JSContext *cx)。

语法 （**Syntax**）

```
Var userModule = loadModule(dllpath);
```

参数 （**Parameters**）

Parameter	Description
dllpath	需要加载动态连接库（DLL）的路径.

返回值 （**Return Values**）

Dll中GetModuleObject方法返回的对象。

示例 （**Example**）

```
loadModule("Sysinfo.dll");
```

1.5 控件风格

通用风格

```
BORDER                  = 0x10000
```

EditBox 风格

```
MULTILINE               = 0x1,
READONLY                = 0x2,
NUMBER                  = 0x4,
VSCROLL                 = 0x8,
HSCROLL                 = 0x16
```

ListBox 风格

```
        MULTISELECTION          = 0x1,
        VSCROLL                 = 0x2,
```

ComboBox 风格

```
        DROPDOWNLIST            = 0x1,
        VSCROLL                 = 0x2,
```

Static 风格

```
        BITMAP                  = 0x1
```

TrackBar 风格

```
        VERT                    = 0x1,
        NOTICKS                 = 0x2,
        AUTOTICKS               = 0x4
```

ProgressBar风格

```
        VERT                    = 0x1,
        SMOOTH                  = 0x2
```

Window 风格

```
        NOCAPTION               = 0x1,
        BORDER                  = 0x2
```

第二章 示例代码

本章节通过两份完整的示例代码，展示如何利用XmlGlue技术进行Widget开发。

示例1

```
<x:elastos xmlns="http://www.elastos.com/XmlGlue/Widgets"
xmlns:x="http://www.elastos.com/XmlGlue/2.1">
        <window text="widget" left="0" top="0" width="240" height="320"
onTimer="OnTimer(timerId)">
                <pushButton x:id="button1" text="quit" left="120" top="250"
width="100" height="30" onSelect="Window.Close()" />
                <pushButton x:id="button2" text="Draw" left="10" top="250"
width="100" height="30" onSelect="testDraw();" />
        </window>
        <x:script>
                <![CDATA[
                var FORM_WIDTH = 170;
                var FORM_HEIGHT = 230;
                var SPEED  =  20;
                var SURFACE_STARTX   =   20;
                var SURFACE_STARTY   =   20;
                var m_nSurfaceW = 170;
                var m_nSurfaceH = 230;
                Grafix = Window.GetGrafix();
                var pat1;
                var pat2;
                function initPat() {
                var img1 = new Image("clearandbright2.png");
                var img2 = new Image("monalisa2.png");
                pat1 = Grafix.CreatePattern(img1);
                pat1.Translate(-SURFACE_STARTX, -SURFACE_STARTY);
                pat2 = Grafix.CreatePattern(img2);
```

```
pat2.Translate(-SURFACE_STARTX, -SURFACE_STARTY);
    }
 function Point(x, y)
 {
 this.x = x;
 this.y = y;
 }
 function drawPolygon(points)
 {
 var n = points.length;
 if (n <= 1) return ;
 Grafix.MoveTo(points[0].x, points[0].y);
 for (var i = 1; i < n; i++) {
 Grafix.LineTo(points[i].x, points[i].y);
 }
 Grafix.ClosePath();
 }
 function testDrawPolygon()
 {
 var points = new Array(4);
 points[0] = new Point(10, 10);
 points[1] = new Point(100, 30);
 points[2] = new Point(70, 90);
 points[3] = new Point(40, 40);
 drawPolygon(points);
 Grafix.Stroke();
 }
 function draw(x, y) {
 if (x > m_nSurfaceW || y > m_nSurfaceH) return;
 Grafix.ClearRect(0, 0, m_nSurfaceW, SURFACE_STARTY);
 Grafix.ClearRect(0, 0, SURFACE_STARTX, m_nSurfaceH);
 Grafix.Save();
 Grafix.pattern = pat1;
```

```
                Grafix.Paint();
                Grafix.Restore();
                if (x != 0 && y != 0 && x != m_nSurfaceW && y != m_nSurfaceH) {
                var tx = SURFACE_STARTX;
                var ty = SURFACE_STARTY;
                tx = (m_nSurfaceW*m_nSurfaceW - y*y - m_nSurfaceH*m_nSurfaceH - x*x+
2*m_nSurfaceH*y)/(2*m_nSurfaceW - 2*x);
                ty = (m_nSurfaceH*m_nSurfaceH - y*y - m_nSurfaceW*m_nSurfaceW - x*x+
2*m_nSurfaceW*x)/(2*m_nSurfaceH - 2*y);
                if (tx <= SURFACE_STARTX) return ;
                var points = new Array(4);
                points[0] = new Point(x, y);
                points[1] = new Point(tx, m_nSurfaceH);
                points[2] = new Point(m_nSurfaceW - Math.abs(ty) * (m_nSurfaceW - tx)
/ (m_nSurfaceH + Math.abs(ty)), SURFACE_STARTY);
                points[3] = new Point(points[2].x - Math.sin(Math.atan2((tx -
x),(m_nSurfaceH - y))) * (m_nSurfaceW - points[2].x), SURFACE_STARTY -
Math.cos(Math.atan2((tx - x), (m_nSurfaceH - y))) * (m_nSurfaceW - points[2].x));
                var triPoints = new Array(3);
                triPoints[0] = new Point(x, y);
                triPoints[1] = new Point(tx, m_nSurfaceH);
                triPoints[2] = new Point(m_nSurfaceW, ty);
                var bgPoints = new Array(4);
                bgPoints[0] = new Point(m_nSurfaceW, m_nSurfaceH);
                bgPoints[1] = new Point(tx, m_nSurfaceH);
                bgPoints[2] = new Point(m_nSurfaceW - Math.abs(ty) * (m_nSurfaceW -
tx) / (m_nSurfaceH + Math.abs(ty)), SURFACE_STARTY);
                bgPoints[3] = new Point(m_nSurfaceW, SURFACE_STARTY);
                var anoPoints = new Array(3);
                anoPoints[0] = new Point(m_nSurfaceW, m_nSurfaceH);
                anoPoints[1] = new Point(tx, m_nSurfaceH);
                anoPoints[2] = new Point(m_nSurfaceW, ty);
                Grafix.fillRule = "winding";
```

```
                Grafix.Save();
                if (ty < SURFACE_STARTY) {
                drawPolygon( points );
                } else {
                drawPolygon( triPoints );
                }
                var pat3 = Grafix.CreateLinearGradient(x, y, m_nSurfaceW, m_nSurfaceH
+ 10);
                pat3.AddColorStop(0, 96, 96, 96);
                pat3.AddColorStop(0.29, 168, 168, 168);
                pat3.AddColorStop(0.68, 0, 0, 0);
                Grafix.pattern = pat3;
                Grafix.Fill();
                Grafix.Restore();
                if (ty < SURFACE_STARTY) {
                                drawPolygon( bgPoints );
                }
                else {
                                drawPolygon( anoPoints );
                }
                Grafix.Save();
                Grafix.pattern = pat2;
                Grafix.Fill(true);
                Grafix.Restore();
                Grafix.Save();
                pat3 = Grafix.CreateLinearGradient((x + m_nSurfaceW)/2, (y +
m_nSurfaceH)/2, m_nSurfaceW, m_nSurfaceH);
                pat3.AddColorStop(0, 0, 0, 0, 0.5);
                pat3.AddColorStop(0.15, 64, 64, 64, 0.05);
                Grafix.pattern = pat3;
                Grafix.Fill();
                Grafix.Restore();
                }
```

```
                }
                var m_x;
                var m_y;
                function OnTimer(id)
                {
                if (id == 1) {
                if ((m_x <= -10) || (m_y <= -10)) {
                var temp =  pat1;
                pat1 = pat2;
                pat2 = temp;
                 m_x = 0;
                m_y = 0;
                Window.KillTimer(1);
                }
                else {
                m_y -= SPEED /2 ;
                m_x -= 2 * SPEED;
                }
                draw(m_x, m_y);
                Grafix.Update();
                }
                }
                function testDraw()
                {
                 m_x = m_nSurfaceW + 1;
                m_y = m_nSurfaceH + 1;
                Window.SetTimer(1, 50);
                }
                initPat();

        ]]>
</x:script>
</x:elastos>
```

示例**2**

```
<?xml version="1.0" encoding="utf-8" ?>
<x:elastos xmlns="http://www.elastos.com/XmlGlue/Widgets"
xmlns:x="http://www.elastos.com/XmlGlue/2.1">
        <window text="widget" left="0" top="0" width="240" height="320">
                <comboBox x:id="option" left="10" top="270" width="100" height="60">
                                <item selected="true" text="test line" />
                                <item text="Image 1" />
                                <item text="Image 2" />
                                <item text="Image 3" />
                                <item text="Image 4" />
                                <item text="Image 5" />
                                <item text="pattern" />
                </comboBox>
                <pushButton x:id="button1" text="quit" left="180" top="250"
width="50" height="40" onSelect="Window.Close()" />
                <pushButton x:id="button2" text="Draw" left="120" top="250"
width="50" height="40" onSelect="OnSelect();" />
        </window>
        <x:script>
                <![CDATA[
                Grafix = Window.GetGrafix();
                img = new Image("romedalen.png");
                pat = Grafix.CreatePattern(img);
                function DrawLine() {
                Grafix.Save();
                Grafix.lineWidth = 10;
                Grafix.SetPattern(155, 155, 155);
                Grafix.lineCap = "round";
                Grafix.lineJoin = "bevel";
                Grafix.MoveTo(10, 10);
```

```
Grafix.LineTo(100, 100);
Grafix.LineTo(190, 10);
Grafix.Stroke();
Grafix.lineCap = "butt";
Grafix.lineJoin = "round";
Grafix.MoveTo(10, 40);
Grafix.LineTo(100, 130);
Grafix.LineTo(190, 40);
Grafix.Stroke();
Grafix.lineCap = "square";
Grafix.lineJoin = "miter";
Grafix.MoveTo(10, 70);
Grafix.LineTo(100, 160);
Grafix.LineTo(190, 70);
Grafix.Stroke();
Grafix.Restore();
Grafix.Save();
Grafix.MoveTo(10, 10);
Grafix.LineTo(100, 100);
Grafix.LineTo(190, 10);
Grafix.MoveTo(10, 40);
Grafix.LineTo(100, 130);
Grafix.LineTo(190, 40);
Grafix.MoveTo(10, 70);
Grafix.LineTo(100, 160);
Grafix.LineTo(190, 70);
Grafix.Stroke();
Grafix.Restore();
}
function DrawImage() {
Grafix.ClearRect(10, 10, 120, 120);
Grafix.DrawImage(img, 10, 10);
}
```

```
function DrawImage2() {
Grafix.ClearRect(10, 10, 50, 64);
Grafix.DrawImage(img, 10, 10, 50, 64);
}
function DrawImage3() {
Grafix.ClearRect(10, 10, 50, 64);
Grafix.DrawImage(img, 20, 30, 50, 64, 10, 10, 50, 64);
}
function DrawImage4() {
Grafix.ClearRect(10, 10, 120, 120);
Grafix.Rect(10, 10, 100, 108);
Grafix.Save();
Grafix.Stroke(true);
Grafix.Restore();
Grafix.Save();
Grafix.Translate(10, 10);
Grafix.pattern = pat;
Grafix.Fill();
Grafix.Restore();
print("end of DrawImage 4");
}
function DrawImage5() {
Grafix.Save();
Grafix.Rect(0, 0, 240, 240);
Grafix.Clip();
Grafix.Translate (120.0, 120.0);
Grafix.Rotate (Math.PI / 4);
Grafix.Scale (1 / Math.sqrt (2), 1 / Math.sqrt (2));
Grafix.Translate (-120.0, -120.0);

var pat2 = Grafix.CreatePattern(img);
pat2.extend = "reflect";
pat2.Scale(256/240.0 * 5.0, 192/240.0 * 5.0);
```

```
            Grafix.pattern = pat2;
            Grafix.Rect(0, 0, 240.0, 240.0);
            Grafix.Fill();
            Grafix.Restore();
            print("end of DrawImage 5");
            }
            function Draw6() {
            print("--------------------------1");
            Grafix.Save();
            var pat2 = Grafix.CreateLinearGradient(0.0, 0.0, 0.0, 240.0);
            pat2.AddColorStop(1, 0, 0, 0, 1);
            pat2.AddColorStop(0, 255, 255, 255, 1);
            Grafix.Rect(0, 0, 240, 240);
            Grafix.pattern = pat2;
            Grafix.Fill();
            Grafix.Update();
            pat2 = Grafix.CreateRadialGradient(112.2, 100.4, 24.6, 100.4, 100.4,
120.0);
            pat2.AddColorStop(0, 255, 255, 255, 1);
            pat2.AddColorStop(1, 0, 0, 0, 1);
            Grafix.Arc(120.0, 120.0, 73.8, 0, 2 * 3.1415926);
            Grafix.pattern = pat2;
            print("--------------------------2");
            Grafix.Fill();
        print("--------------------------3");
            Grafix.Restore();
            print("end of Draw 6");
            }
            function OnSelect()
            {
            selection = option.GetCurrentSelection();
            if (selection == 0) {
            DrawLine();
```

```
                }
                else if (selection == 1) {
                DrawImage();
                }
                else if (selection == 2) {
                DrawImage2();
                }
                else if (selection == 3) {
                DrawImage3();
                }
                else if (selection == 4) {
                DrawImage4();
                }
                else if (selection == 5) {
                DrawImage5();
        }
        else if (selection == 6) {
                Draw6();
                }
                Grafix.Update();
        }
                ]]>
        </x:script>
</x:elastos>
```

索引