XML UI Language

**v0.1**

|  |
| --- |
| **2** |

|  |  |  |  |
| --- | --- | --- | --- |
| **拟 制** | 黄毅 | **日期** | 2014-4-30 |
| **评审人** |  | **日期** |  |
| **批 准** |  | **日期** |  |

**修订记录**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **日期** | **修订版本** | **修改章节** | **修改描述** | **作者** |
| 2014-04-30 | 0.1 |  | 创建文档 | 黄毅 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[XML UI Language 1](#_Toc405196568)

[目录 3](#_Toc405196569)

[1 概述 5](#_Toc405196570)

[1.1 目的 5](#_Toc405196571)

[2 基本结构 5](#_Toc405196572)

[2.1 示例 5](#_Toc405196573)

[2.2 数据分离 6](#_Toc405196574)

[3 XUL设计 7](#_Toc405196575)

[3.1 基本节点 9](#_Toc405196576)

[3.1.1 starcor.xul 根节点 9](#_Toc405196577)

[3.1.2 page UI页面 9](#_Toc405196578)

[3.1.3 layout UI页面布局 10](#_Toc405196579)

[3.1.4 area UI元素分组 10](#_Toc405196580)

[3.1.5 template UI元素 11](#_Toc405196581)

[3.1.6 item UI元素 11](#_Toc405196582)

[3.1.7 公共属性 11](#_Toc405196583)

[3.2 属性、行为、样式、焦点 12](#_Toc405196584)

[3.2.1 attr 属性 12](#_Toc405196585)

[3.2.2 action 行为 12](#_Toc405196586)

[3.2.3 data 用户数据 14](#_Toc405196587)

[3.2.4 style 样式 14](#_Toc405196588)

[3.2.5 focus 焦点控制 16](#_Toc405196589)

[3.3 select 选择器 19](#_Toc405196590)

[3.3.1 基本用法 19](#_Toc405196591)

[3.3.2 嵌套选择(暂不支持) 20](#_Toc405196592)

[3.3.3 优先级 21](#_Toc405196593)

[3.4 数据绑定 22](#_Toc405196594)

[3.4.1 通过选择器绑定数据 22](#_Toc405196595)

[3.4.2 通过动态数据绑定机制绑定数据 22](#_Toc405196596)

[3.5 URL解释 27](#_Toc405196597)

[4 图片与动画 27](#_Toc405196598)

[4.1 支持的图片格式 27](#_Toc405196599)

[4.2 Nine-Patch图片 28](#_Toc405196600)

[4.3 帧动画 28](#_Toc405196601)

[5 扩展组件 29](#_Toc405196602)

[5.1 Label组件 29](#_Toc405196603)

[5.2 Image组件 30](#_Toc405196604)

[5.3 Slider组件 31](#_Toc405196605)

[5.4 Page Slider组件 32](#_Toc405196606)

[5.5 Grid组件 32](#_Toc405196607)

[5.6 Group/Radio/Check组件 32](#_Toc405196608)

[5.7 Custom组件 33](#_Toc405196609)

[6 XUL脚本 34](#_Toc405196610)

[6.1 概念 34](#_Toc405196611)

[6.2 对象 34](#_Toc405196612)

[6.2.1 XUL对象 34](#_Toc405196613)

[6.2.2 logger对象 34](#_Toc405196614)

[6.2.3 View对象 35](#_Toc405196615)

[6.2.4 Area对象 38](#_Toc405196616)

[6.2.5 Slider对象 38](#_Toc405196617)

[6.2.6 PageSlider对象 38](#_Toc405196618)

[6.2.7 Group对象 39](#_Toc405196619)

[6.2.8 Page对象 39](#_Toc405196620)

[7 应用实例 41](#_Toc405196621)

# 概述

## 目的

XML UI Language描述了一种XML方言。通过XML格式数据对UI的布局、元素、样式、行为、数据绑定关系等进行描述。实现了一种简单、易用、可扩展的动态UI渲染机制。

# 基本结构

## 示例

如下：

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

<starcor.xul>

<page id="main">

<layout id="main">

<item type="label">

<attr name="text">hello world</attr>

</item>

</layout>

</page>

</starcor.xul>

这是一个简单的UI描述，是一个标准的XML文档。使用starcor.xul节点作为根级元素。

其中，page节点描述了一个独立的UI页面。它必须放在starcor.xul节点下。layout节点用来描述页面的布局，其中可以包含其它的UI元素。这里使用了一个类型（type）为“label”的item元素，其文本属性的值为“hello world”。

hello world

其显示效果如上所示。

## 数据分离

在前面的示例中，UI元素内部直接绑定了数据属性。为了达到数据与布局分离的目的，可以通过选择器选定元素并设置其属性。如下：

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

<starcor.xul>

<page id="main">

<layout>

<item id="text" type="label"/>

</layout>

</page>

<selector>

<select id="text">

<attr name="text">hello world</attr>

</select>

</selector>

</starcor.xul>

这个例子与前例效果完全相同。但文本标签的数据是通过select节点在后期绑定的。

select节点可以通过元素的id，type，class，state进行选择。可以选择一个元素，也可以一次选中多个元素。

元素（item）、区块（area）、页面（layout）的属性、行为、样式等参数都可以通过选择器进行设置。

除了使用选择器外，通过动态数据绑定机制也可以实现数据与布局的分离。如下：

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

<starcor.xul>

<page id="abc">

<binding>

<label1>

<text>hello world</text>

</label1>

</binding>

<layout>

<item type="label">

<attr name="text" binding="label1/text"/>

</item>

</layout>

</page>

</starcor.xul>

attr，binding，data，focus，style，template可以通过binding属性设置为动态数据绑定模式。在此模式下，节点的值/数据源通过binding属性选择的数据填充/指定。当前页面的动态数据源通过binding节点指定，可以是URL或XML数据字符串。

# XUL设计

|  |  |  |  |
| --- | --- | --- | --- |
| **元素** | **直接父级** | **直接子级** | **说明** |
| action | area, layout, select, item, page, component | 任意内容 | 行为 |
| area | area, layout, template, component | action, area, attr, item, focus, style, template, data | 元素分组 |
| attr | area, layout, select, item, page, component | 任意内容 | 属性 |
| binding | page, starcor.xul | URL, XML数据, json数据 | 动态数据绑定 |
| bottom | focus | select | 下方相邻焦点 |
| component | starcor.xul | area, attr, data, item, template, focus, selector, style, action | 组件 |
| focus | area, layout, select, item, component | left, right, top, bottom | 焦点 |
| import | starcor.xul | 无 | 引用其它xul文件 |
| item | area, layout, component | action, attr, data, style, focus | 基本界面元素 |
| layout | page | area, attr, data, item, template, focus, style | UI页面布局 |
| left | focus | select | 左方相邻焦点 |
| page | starcor.xul | attr, action, layout, selector, binding | UI页面根级元素 |
| right | focus | select | 右方相邻焦点 |
| script | starcor.xul | 无 | 引用脚本文件 |
| selector | starcor.xul, page, component | select | 选择器集合 |
| starcor.xul | 无 | page, selector, import, binding, script, component | 根级元素 |
| style | area, layout, select, item, component | 任意内容 | 样式 |
| template | area, layout, component | attr, area, item | 布局模板 |
| top | focus | select | 上方相邻焦点 |

## 基本节点

### starcor.xul 根节点

所有的元素必须为此节点的直/间接子级。

根节点下可以包含：

1. page 页面节点
2. selector 选择器集合
3. import 外部引用
4. binding动态数据源
5. script 外部脚本引用
6. component组件

**可用属性**

|  |  |
| --- | --- |
| 属性 | 说明 |
| screen | 设置全局默认的屏幕分辨率（如：1280x720）。当实际屏幕分辨率与此参数不符时，自动进行缩放。**必选参数** |
| xmlns | 设置xml命名空间，一般使用  "http://www.starcor.cn/XUL" |
|  |  |

### page UI页面

page是UI渲染的基本单位。所有其它元素都不能直接渲染，必须放在page/layout中才可以被渲染。

page中可以包含layout及selector节点：

1. layout节点用于描述页面的内容布局。*当有page有多个layout时，每个layout必须使用不同的screen属性。页面根据screen参数选择合适的布局。(暂不实现)*
2. selector节点用于存放页面私有选择器（select）。

**可用属性**

|  |  |
| --- | --- |
| 属性 | 说明 |
| screen | 设置页面默认的屏幕分辨率（如：1280x720）。当实际屏幕分辨率与此参数不符时，自动进行缩放。无此参数时使用全局默认分辨率参数。 |
|  |  |

### layout UI页面布局

layout代表了一个页面的布局。

*一个页面可以有多种不同的布局，这些布局根据screen参数自动适配。（暂不支持）*

**可用属性**

|  |  |
| --- | --- |
| 属性 | 说明 |
| screen | 设置当前布局默认的屏幕分辨率（如：1280x720）。当实际屏幕分辨率与此参数不符时，自动进行缩放。无此参数时使用页面默认分辨率参数。 |
| x，y | 设置layout的显示位置，默认为0，0。 |
| width，height | 设置layout的尺寸，默认与屏幕尺寸相同。 |
|  |  |

### area UI元素容器

area元素代表了元素（焦点）的容器。通过area元素容器可以改变焦点移动的行为，也可以改变元素的显示效果（滚动，分页等）。

**可用属性**

|  |  |
| --- | --- |
| 属性 | 说明 |
| x，y | 设置area的显示位置，默认为auto，auto。 |
| width，height | 设置area的尺寸，默认为auto，auto。 |
|  |  |

|  |  |
| --- | --- |
| 样式 | 说明 |
| preload | 设置元素强制预加载，所有子级元素都会预加载。默认为false |
| keep-focus-visible | 强制焦点不能移动出可见范围 |
|  |  |

### component 用户UI组件

用户组件用于定义一组预定义的UI布局片断。可以通过id引用实例化。它可以在不同的页面中共享相同的布局样式。

### template UI元素

template元素用来表示一类数据呈现的局部布局模板。只有绑定了数据的模板才能被渲染出来。当template绑定了多份数据时，模板会被渲染多次。

**可用属性**

|  |  |
| --- | --- |
| 属性 | 说明 |
| binding | 设置template绑定的数据。 |
| incremental | 设置template为增量式数据绑定模式。默认为false。  此模式将向父级元素发送incrementalBindingUpdate / incrementalBindingFinished事件  格式：  true/enabled 等价于 1, 200  delay-time  pre-create-num, delay-time |

template可以有一个或多个子元素。子元素可以是area、item或template。

每个子元素可以通过filter属性对绑定的数据进行过滤，仅在指定的条件下被实例化。

<template binding="xxxx">  
 <item type="image" filter="[type=image]"></item>  
 <item type="label" filter="[type=label]"></item>  
</template>

如上例所示：

Image元素仅在绑定的数据type属性值为image时实例化，而label元素则仅在绑定数据type属性为label时实例化。如果有多个元素的filter满足条件，则仅第一个元素被实例化。

Filter语法如下所示：

|  |  |
| --- | --- |
| Filter语法 | 说明 |
| [属性=值]  [属性] | 属性过滤器  [type=image]表示type属性值为image的数据  [type]表示有type属性的数据 |
| [n]  [x,y,z]  [x~y]  [x~]  [xN+m] | 位置过滤器（从0开始）  [-1]表示过滤最后一个数据  [1,2,3]表示过滤第1,2,3个数据  [1~8]表示第1到8个数据，闭区间  [2N+1]表示索引为奇数的数据 |

### item UI元素

item为最基本的UI元素。可以用来显示图片，文字等内容。只有item可以获取焦点。

**可用属性**

|  |  |
| --- | --- |
| 属性 | 说明 |
| x，y | 设置item的显示位置，默认为auto，auto。 |
| width，height | 设置item的尺寸，默认为auto，auto。 |
|  |  |

### script 节点

script节点用于声明对外部脚本文件的引用。如下：

<script type="script/javascript" src="file:///.assets/test.js"/>

所有的脚本加载到全局上下文中。后续action可通过脚本中的函数名引用脚本文件代码。

### 公共属性

|  |  |
| --- | --- |
| 属性 | 说明 |
| id | 元素ID，应全局唯一。注意：模板元素在多次实例化时，可能造成多个元素使用同一个ID。 |
| class | 元素分类标识。用户自定义，多个标识通过“，”分隔。 |
| type | 元素类型。用于指定元素的行为特征。 |
|  |  |

## 属性、行为、样式、焦点

### attr 属性

attr为属性节点。通过属性节点可以对item，area，layout以及component进行设置，修改其属性值。如下：

<item>

<attr name="height">120</attr>

</item>

将item的height属性设置为120。

通过attr可以修改**除**id、class、type外的所有其它元素属性。

|  |  |
| --- | --- |
| 属性 | 说明 |
| x | 元素x坐标 |
| y | 元素y坐标 |
| width | 元素宽度 |
| height | 元素高度 |
| enabled | 是否可用，true/false， 默认为true |
| animation | 启用动画。enabled/disabled，默认为enabled |
| animation-duration | 动画时长。单位ms |
| animation-mode | 动画模式：mode [: param1 [, param2 [, …] ] ]  模式如下：  linear（线性，默认值）y = x  bouncing弹跳，参数：speed, freq, slope, strength  bouncing-back回弹，参数：slope,speed,strength  sin，参数：slope |  pow，参数：e |  shaking，参数：freq, offset, delta, strength, grow  constant（匀速），参数：speed, start-duration, end-duration  speed参数为px，表示一个动画周期内移动多少px  start-duration为启动时间，默认为0ms  end-duration为结束时间，默认为0ms |
| animation-sizing  animation-moving | 元素width/height变换动画，默认为false  元素x/y变换动画，默认为false |
|  |  |

### action 行为

action节点用于定义元素的事件-动作绑定关系。如下：

<item>

<action event="click" type="usr\_cmd">xxxx</action>

</item>

此例中event属性定义了action的触发事件。type属性定义了action绑定动作的类型。不同的动作类型决定了动作执行的方式。

通过这个action节点，我们将item的click事件绑定到类型为usr\_cmd的动作xxxx上。当item被点击时，其对应的动作将被触发。

可用事件列表如下：

|  |  |
| --- | --- |
| **事件** | **说明** |
| click | 元素被点击 |
| focus | 元素获得焦点 |
| blur | 元素失去焦点 |
| checked | 元素被设置为选中状态 |
| unchecked | 元素被设置为未选中状态 |
| timer | 定时器事件，可以通过参数设置定时器间隔。（暂未实现）  如：timer(1s)表示每秒触发一次  可用的单位：毫秒(ms)、秒(s)、分钟(m)、小时(h)、天 (d)  timer(1d2m3ms) - 1天零2分零3毫秒 |
| key  keydown  keyup | 按键事件，如：（暂未实现）  key接收所有按键事件  key(back) 接收返回键  key(back+alt+ctrl-shift) 接收返回键，且按下alt，ctrl松开shift  key(dpad\_center,enter) 接收dpad\_center以及enter事件  keydown(back) 接收返回键按下事件  keyup(4) 接收键值为4的按键弹起事件 |
| text | 文本输入事件（暂未实现） |
| bindingFinished | 元素数据绑定已经完成 |
| bindingUpdated | 元素数据绑定已经更新 |
| incrementalBindingUpdate | 增量数据绑定更新事件（仅template组件的父级容器可收到此事件）  参数：  arguments[0] ：template组件的id  arguments[1] ：当前绑定数据索引  arguments[2] ：总数据条目数  arguments[3] ：当前创建的对象，对象创建失败时此值为空 |
| incrementalBindingFinished | 增量数据绑定完成事件（仅template组件的父级容器可收到此事件）  参数：  arguments[0] ：template组件的id |
| bindingReady | 绑定数据源加载成功，仅页面元素有效。  对于间接绑定数据源无效。  参数：  arguments[0] ：binding节点id |
| bindingError | 绑定数据源加载失败，仅页面元素有效。  对于间接绑定数据源无效。  参数：  arguments[0] ：binding节点id  arguments[1] ：错误原因 |
| load | 页面布局加载成功，仅页面元素有效 |
| ready | 页面所有数据绑定完成 或 元素布局/属性初始化完成 |
| statesRestored | 页面状态还原。仅由back按键触发，用户手工调用popStates还原页面状态不会触发此事件。 |
| loadImageSuccess  loadImageFailed | Image组件加载图层成功/失败事件  参数：  arguments[0] ：image图层编号  arguments[1] ：加载所用时间（ms） |
| pageChanged | page\_slider页面切换事件 |
| indicatorChanged | slider滚动指示器状态变化事件。  参数：  arguments[0] ：指示器标识（right,left,top,bottom）  arguments[1] ：显示状态（true/false） |
| componentInstanced | component实例化完成事件 |
| componentChanged | component属性变化事件 |
| scrollStopped | slider组件滚动停止事件  参数：  arguments[0] ：当前位置（scrollPos）  arguments[1] ：最大滚动范围（scrollRange） |

可用动作列表如下：

|  |  |  |
| --- | --- | --- |
| **类型** | **动作** | **说明** |
| usr\_cmd | 用户自定义命令，由应用自行扩充 | |
|  |  |
|  |  |
| script/javascript | 执行javascript脚本  action命令可以是全局脚本中的函数名，或脚本代码。 | |
|  |  |
|  |  |

### data 用户数据

为避免属性冲突，所有的用户数据都应该保存在data属性中。此属性与attr结构类似。如下：

<item>

<data name="userdata">userdata</data>

</item>

用户数据的格式根据应用场景由用户自行定义。所有的用户数据在action被触发时自动传递给动作相应的处理逻辑。

### style 样式

通过style节点可以设置元素的UI样式。如下：

<item>

<style name="left">10</style>

<style name="top">20</style>

</item>

可用的样式如下：

|  |  |
| --- | --- |
| 样式 | 说明 |
| left，top，right，bottom | 元素位置，默认值为auto（暂不实现） |
| width，height | 元素尺寸。（暂不实现）  *当元素位置与元素尺寸同时存在时，后设置的有效。* |
| padding | 元素内边距  top，left，right，bottom  all  top/bottom，left/right  可通过以下名称分别设置在不同方向的内边距  padding-left，padding-right，padding-top，padding-bottom  指定了方向的内边距样式（如：padding-left）会覆盖padding设置的参数。 |
| margin | 元素外边距（仅对slider, grid组件的子元素有效）  可通过以下名称分别设置在不同方向的外边距  margin-left，margin-right，margin-top，margin-bottom |
| position | 元素定位模式（暂不实现）  absolute 绝对定位。默认值。  static 静态定位。使用流式布局。 |
| preferred-focus-padding | 设置优先焦点边距，即焦点四边到容器有效焦点区域的最小距离。格式同padding  焦点元素  父容器边距  焦点边距  在条件允许的情况下，尽可能把子元素焦点移动到符合padding限制的位置显示。仅对slider有效。 |
| display | 元素显示模式  block显示。默认。  none 隐藏。 |
| scale | 元素缩放。  scalar，[x-alignment，y-alignment]  scalar-x，scalar-y，[x-alignment，y-alignment]  scalar为元素缩放比例，如：1.2表示放大20%  x-alignment，y-alignment表示缩放对齐模式，可选值为：  left，top，right，bottom，v-center，h-center，center或对齐位置的值。默认值为center。  如：1.2，left，v-center表示放大20%，水平左对齐，上下居中对齐。其等价值形式为 1.2，0，0.5。  *行/行块模式对齐时，水平对齐参数将不起任何作用！* |
| animation-scale | 启用元素缩放动画。enabled/disabled，默认为enabled |
| font-size  font-color  font-weight  font-scale-x  font-shadow  font-align  font-resample  font-style-italic  font-style-underline  font-style-strike  font-face | font-size 数字  font-color 十六进制颜色  font-weight 浮点数，大于1表示粗体  font-scale-x 浮点数，默认为1.0  font-shadow 格式x, y, size, color  font-align 格式x, y。取值范围0~1  font-resample 浮点数，默认为1.0文本超采样比例  font-style-italic 布尔值，true/false  font-style-underline布尔值，true/false  font-style-strike 布尔值，true/false  font-face字符串，设置对象使用的字体。字体必须通过XulRenderContext.addTypeFace(“名称”, <字体对象>)手工添加。 |
| font-render | 文本渲染模式, simple/spannable 默认为simple  对label/image有效 |
| start-indent  end-indent | 文本缩进, 数字, 单位像素  start-indent首行行首缩进, 仅对多行文本有效  end-indent末行行尾缩进, 仅对multi-line, auto-wrap, ellipsis文本有效 |
| fix-half-char | 避免显示半个字符，true/false，默认为false（label、image组件适用） |
| line-height | 文本行高，默认为1.0（100%）。 |
| background-image  background-color |  |
| border | 元素边框。  宽度，颜色，圆角x-radius，圆角y-radius，位置  宽度，颜色，位置  如：8,E01595EF,12,12,0.35 |
| border-dash-pattern | 元素边框模式。可以实现各类虚线效果。  pattern\_val1, pattern\_val2  phase, pattern\_val1, pattern\_val2[, pattern\_valN…] |
| clip-children  clip-focus | 绘制时是否裁减子级元素（true/false）  绘制时是否裁减焦点元素（true/false） |
| z-index | 元素的绘制层级。值越大，层级越高。层级高的元素覆盖层级低的元素。 |
| do-not-match-text | 元素match\_content时不考虑text的尺寸，仅image组件有效 |
| layout-mode | label, image组件有效  ltr 默认值，从左向右布局  rtl 从右向左布局，自动翻转x对齐方式及左/右padding。 |
| quiver | 元素抖动效果，参数：  dx,dy  dx,dy,duration  dx,dy,duration,repeat  dx,dy,duration,repeat,repeat-strength  当dx、dy绝对值大于等于1.0时，表示抖动偏移象素。小于1.0时表示百分比偏移。  repeat 定义元素抖动循环次数, 默认为1  repeat-strength定义元素循环强度变化比例, 默认为1 |
| quiver-mode | 指定quiver效果动画算法，格式请参考animation-mode属性。默认值为pow:0.15 |
| min-width  min-height  max-width  max-height | 限定元素最小/大的宽度和高度。  仅对自动宽高元素有效。 |
| lock-focus-target | 将元素标记为焦点锁定目标 |

### focus 焦点控制

focus节点用来设置元素的焦点行为。如：

<layout>

<area>

<!-- 设置area焦点模式为loop -->

<focus mode="loop"/>

<item>

<!-- 设置当前item为焦点 -->

<focus focused="true"/>

</item>

<item />

</area>

</layout>

#### mode设置焦点模式

可用模式

|  |  |
| --- | --- |
| 模式 | 说明 |
| nearby | 邻近模式。焦点移动时，优先选择最近的焦点元素。**默认值**。 |
| priority | 优先级查找。**默认值**。 |
| dynamic | 动态模式。区块中的焦点元素使用动态优先级，上次的焦点元素具有最高优先级。 |
| loop | 循环模式。区块中的焦点自动循环(仅slider, grid支持)。 |
| wrap | 自动换行模式。区块中当前方向无目标焦点时，自动查找下一行焦点元素。（仅grid支持） |
| nofocus | 无焦点模式，元素不可获取焦点。area默认为此模式。 |
| focusable | 有焦点模式，元素可获取焦点。item默认为此模式。 |

模式前加“~”表示去除此模式，如：~nearby表示不进行邻近焦点查找。

#### 焦点查找顺序

nearby, priority, dynamic三种模式的焦点查找顺序根据其值出现在mode属性中的顺序执行。默认按 dynamic, priority, nearby的顺序查找。

#### focused设置当前焦点

当focused为true时，将item，area设置为焦点状态。整个页面只能存在一个焦点；如果有多个元素分别被设置成焦点，则**第一个**有效。

#### priority设置焦点优先级

焦点移动时，如果当前元素在移动方向上找不到可用焦点，则会从相邻分组中选择焦点优先级高（大于0的）的元素作为新焦点。

如果没有可用高优先级元素，则按当前焦点规则从所有可用目标中选择新焦点。所有元素的默认焦点优先级为-1。

如下：

#### left，top，right，bottom 设置相邻焦点(未实现)

设置item/area相邻焦点。如下：

<area id="left-area">

<focus mode="loop"/>

<item type="xxxx"/>

<item type="xxxx"/>

<item type="xxxx"/>

<item type="xxxx"/>

</area>

<area>

<!-- 设置area的模式为nearby，并绑定其左方的邻近焦点 -->

<focus mode="nearby">

<left>

<select id="left-area">

<select target="children" type="xxxx" nth="0"/>

</select>

</left>

</focus>

<item>

<focus focused="true"/>

</item>

</area>

在focus节点中，通过left绑定选择器，选择area的左相邻焦点。当焦点移动时，会自子级向父级逐级查找符合条件的焦点元素。当查找到area区块时，将使用它绑定的选择器选择新的焦点。如果选择器返回了多个目标焦点，则使用第一个目标作为新的焦点。

## select 选择器

select节点实现了选择器的功能。通过选择器可以对单个或多个元素进行修改，实现数据绑定等操作。

### 基本用法

#### 通过id选择元素

<select id="item-id">

<attr name="text">new value</attr>

</select>

如上所示，select选择了id为“item-id”的元素，并将其“text”属性设置为“new value”。

#### 通过class选择元素

<select class="nofocus">

<focus mode="nofocus"/>

</select>

如上所示，通过select将class为“nofocus”的元素设置为无焦点模式。

#### 通过type选择元素

<select type="label">

<focus mode="nofocus"/>

<style name="font-size">12pt</style>

</select>

如上所示，通过select将type为“label”的元素设置为无焦点模式，字体大小为12pt。

#### 通过state选择元素

<select type="label" state="focused">

<style name="scale">1.2</style>

</select>

如上所示，通过select将具有焦点的label元素放大20%。

可用的状态有，focused，disabled。

***注意：focus节点不应通过state选择器绑定。绑定focus节点时会忽略state选择器。***

***注意：通过state选择元素必须结合其它属性（id、type、class）才可以进行选择***

#### 通过位置选择元素（暂不支持）

<select class="item" nth="0">

<focus focused="true"/>

</select>

如上所示，通过select将第0个“item”设置为焦点状态。

如果nth的值为负数，则从最后一个元素开始反向计数。如“-1”表示选中最后一个元素；“-2”表示选中倒数第二个元素。

nth值为2n+1时，表示选中结果集中奇数位置的元素；为2n时，表示选中结果集中偶数位置的元素。其它如3n+1，4n+2依次类推。

nth可以指定多个目标，目标用逗号分隔。如：nth=“1,3,5,7”。

### 嵌套选择(暂不支持)

<select id="abc">

<select target="children" class="button"/>

<select target="children" class="image"/>

<style name="padding">24</style>

</select>

如上所示，通过select选择id为“abc”的元素的子元素中，class为“button”***或***“image”的元素，设置其“padding”样式为24。

select可以多级嵌套，子级的select表示对父级选择器的结果进行处理。如果有多个子级，则表示对父级选择器的结果分别进行不同的处理，取其并集。

target属性只能在内嵌select中出现。表示相对于父结果集中元素某一位置上的（相邻/子级/父级）元素。可用的目标有：

|  |  |
| --- | --- |
| 目标 | 说明 |
| children | 所有子级元素 |
| parent | 父级元素。 |
| next | 下一个相邻元素。 |
| prev | 上一个相邻元素。 |

### 优先级

在通过选择器修改元素的属性（或样式、焦点行为）时。存在如下优先级关系。

1. 内联属性优先级最高。如下：

<starcor.xul>

<page>

<layout>

<item id="123">

<attr name="height">32</attr>

</item>

</layout>

</page>

<selector>

<select id="123">

<attr name="height">33</attr>

</select>

</selector>

</starcor.xul>

通过select无法修改元素内联属性，item的height属性仍为32。

1. 多个选择器设置相同元素的同一属性时，根据选择器的位置决定属性的优先级。其优先级按其出现的位置依次增大。如下：

<starcor.xul>

<page>

<layout>

<item id="123"/>

</layout>

</page>

<selector>

<select id="123">

<attr name="height">33</attr>

</select>

<select id="123">

<attr name="height">32</attr>

</select>

</selector>

</starcor.xul>

两个select分别设置item的height，后出现的覆盖先出现的。所以，这里item的height应该为32。

1. 通过动态数据绑定设置的元素属性与内联属性具有相同优先级。
2. 局部选择器优先级高于全局选择器。
3. 选择器的限定条件越多，优先级越高。

## 数据绑定

### 通过选择器绑定数据

使用选择器可以修改指定元素的属性，也可以修改其数据。如下：

<page>

<layout>

<item id="abc" type="label"></item>

</layout>

</page>

<selector>

<select id="abc">

<attr name="text">hello world</attr>

</select>

</selector>

使用选择器绑定数据在使用上有很高的灵活性，但要求数据提供者了解布局的结构。对布局的设计有较高的依赖性。布局变更会影响数据的结构。在绑定复杂嵌套数据时逻辑上较为复杂。

### 通过动态数据绑定机制绑定数据

为了简化数据拼装过程，这里提供了另一种对数据提供者更友好的动态数据绑定机制。如下：

<page id="abc">

<binding>

<label1>

<text>hello world</text>

</label1>

</binding>

<layout>

<item type="label">

<attr name="text" binding="label1/text"/>

</item>

</layout>

</page>

通过在attr节点上添加binding属性，将attr变成动态数据绑定模式。其值由当前页面的数据源提供。当前页面的数据源则通过binding节点指定。**数据源只能在page节点和starcor.xul节点内出现**。

binding既可以直接指定数据实体，也可以通过URL指定文件数据源。比如：

<binding>http://host/helloworld.xml</binding>

#### 数据源

binding节点在动态数据绑定机制中起到数据源的作用。它为数据选择器提供数据。数据的格式可以为XML或JSON。

数据源有以下三种形式：

1. **内联数据**

所有数据直接写在binding节点中，如下：

XML数据

<binding>

<data>abc</data>

</binding>

JSON数据

<binding>

{"key":"val"}

</binding>

1. **文件数据**

在数据源中指定数据的URL地址。如下：

HTTP数据源

<binding>http://host/path</binding>

本地文件数据源

<binding>file:///path/file</binding>

相对地址，相对于布局文件的URL。（暂未实现）

<binding>/path/file</binding>

<binding>./path/file</binding>

数据源可以通过id属性命名，以供数据选择器使用。如下：

<binding id="ds1">/path/file</binding>

<binding id="ds2">./path/file</binding>

详细说明参考 [URL解释](#_URL解释) 章节

1. **间接数据源**

<binding binding="#ds1/path" />

间接数据源可以引用其它的数据源的子节点中的数据。用于简化数据选择器结构，为数据绑定机制带来更多的灵活性。

#### 数据选择器

attr（或其它节点）中的binding属性被称为数据选择器，它使用如下语法：

|  |  |
| --- | --- |
| 语法 | 说明 |
| / | 根级元素 |
| label1/text | label1节点下的text节点 |
| item[0] | 第0个item节点，如果索引为负值表示从最后一个节点开始计数。如item[-1]表示最后一个item节点。 |
| item[0,1,2] | 第0,1,2个item节点，等价于item[0~2]。如果同一索引出现多次则自动合并。 |
| item[1~] | 除了第0个item外的其它item节点 |
| item[name=xx] | name属性为“xx”的item节点 |
| item[=name] | 读取item的name属性 |
| item<value> | 内容为value的item节点  当value：  形如 =regexp=时使用正则表达式匹配内容  形如~regexp~时使用正则表达式忽略大小写匹配内容 |
| :parent  :next  :prev | 父级节点  下一个兄弟节点  上一个兄弟节点 |
| #ds1/item | 数据源ds1下的item元素 |
| {#ds1/item} | 表示间接数据选择器。它使用数据源ds1的item节点的内容作为数据选择器。  比如：当“#ds1/item”的内容为“/abc”时，此项的结果为当前数据源根节点下的abc节点。  间接数据选择器支持连接和嵌套，如：{{#ds1/item}/def }/xyz⬄{/abc/def}/xyz⬄xxx/xyz |
| =? | 赋值选择器。即用“?”处的选择器的值作为选择器的结果。如：={/item}-value，当/item的值为“test”时，节点绑定的数据为“test-value”。**注意：这里的{}不可嵌套。** |
| /abc/item[0] | 所有abc下的第0个item，节点数目最多和abc的数目相同。 |
| (/abc/item)[0] | 所有abc下item的第0个，最多只有一个节点。 |

**所有蓝色数据选择器暂不支持**。

#### JSON数据与XML数据的对应关系

JSON数据在使用时会被映射成XML数据，以便进行数据选择。如下：

|  |  |  |
| --- | --- | --- |
| JSON | XML | 说明 |
| {“key”: “value”} | <object key=”value”>  </object> | 所有简单属性（string,number,boolean,null）会被映射成XML节点属性。  根级对象名称为空（此处object仅用做示意）。 |
| [1,2] | <array>  <0>1</0>  <1>2</1>  </array> | JSON数组所有子元素会被映射成为XML子节点，节点名为元素的索引（从0开始）。 |
| {“key”:{“key2”: “value”}} | <object>  <key key2=”value”>  </key>  </object> | 复杂属性（属性的值为对象，数组）会被映射成为子节点，节点名为属性名。 |
| {  “attr1”:null,  “attr2”:12.34,  “attr3”:false,  “node1”: {  “k1”:”v1”  },  “node2”:[  1,  {“k”:”v”},  [  “a”,  ”b”  ]  ]  } | <object  attr1=(NULL)  attr2=”12.34”  attr3=”false”  >  <node1  k1=”v1  />  <node2>  <0>1</0>  <1 k=”v”/>  <2>  <0>a</0>  <1>b</1>  </2>  </node2>  </object> |  |
|  |  |  |

结点映射示例：

{

"attr1": null,

"attr2": 12.34,

"attr3": false,

"node1": {

"k1": "v1"

},

"node2": [

1,

{

"k": "v"

},

[

"a",

"b"

]

]

}

映射为如下对象结构

<object

    attr1=(NULL)

    attr2="12.34"

    attr3="false"

    >

    <node1 k1="v1"/>

    <node2>

        <0>1</0>

        <1 k="v"/>

        <2>

            <0>a</0>

            <1>b</1>

        </2>

    </node2>

</object>

#### 数据上下文

数据选择器在选择数据时，如不指定数据源则使用当前上下文的默认数据源。 当前上下文的数据源通过使用binding属性设置；如未设置则继承父级元素的默认数据源。如下：

<page id="abc">

<binding>

<userdata>userdata</userdata>

<label1>

<label1>

<text>hello world</text>

</label1>

</label1>

</binding>

<layout>

<data name="userData" binding="userdata"/> <item type="label" binding="lable1">

<!-- 使用item的数据绑定上下文 -->

<attr name="text" binding="label1/text"/>

</item>

</layout>

</page>

#### 模板的动态数据绑定

使用动态数据绑定机制，可以方便的进行模板实例化。如下：

<page id="abc">

<binding>

<item>

<label1>

<text>hello world1</text>

</label1>

</item>

<item>

<label1>

<text>hello world2</text>

</label1>

</item>

<item>

<label1>

<text>hello world3</text>

</label1>

</item>

</binding>

<layout>

<!-- 绑定所有的item节点 -->

<template binding="item">

<item type="label">

<attr name="text" binding="label1/text"/>

</item>

</template>

</layout>

</page>

在template上绑定一组数据，可以使template实例化多次。

## URL解释

XUL中引用到的所有外部资源都统一通过URL进行解释。如：数据绑定时的数据源，UI组件引用的图片等。

URL有如下类型：

1. FILE资源

文件资源由本地应用进行解释，并实现资源映射。格式如下：

<file://path>

path如果以“/”开头用于直接文件路径映射。

比如：<file:///xxx/abc.jpg> => 文件资源基路径/xxx/abc.jpg

path如果以“/.xxx/”形式开关用于表示特殊数据映射。

比如：<file://.app/api/n36>

.app表示读取应用数据，api/n36路径表示读取n36 API数据。

与此类似的还有

.assets表示应用内建媒体资源数据。

.sdcard表示sd卡上的媒体资源数据。

1. HTTP资源

http资源表示标准的HTTP数据地址。

# 图片与动画

## 支持的图片格式

视系统情况而定，可以在xul中使用不同格式的图片。通常在android系统中支持png及jpeg格式的图片。部分支持svg图片，但资源占用较大。

## Nine-Patch图片

xul中可以直接使用android中定义的nine-patch格式的图片，以较少的资源实现高品质的图片缩放显示。

系统通过文件名识别图片是否为nine-patch格式。以.9及.9.png结尾的文件名都被识别为nine-patch格式。图片可以使用编译过的nine-patch图片（仅android平台支持），也可以使用未编译的原始nine-patch图片。

## 帧动画

在xul中支持自定义帧动画渲染。帧动画文件是一个自定义内容的zip压缩包，由一张/组图片及相应的动画描述元数据组成。帧动画的文件名必须以.ani或.ani.zip结尾。

比如：

test.ani.zip由

0.png

ani.txt

两个文件组成。

其中0.png为动画中用到的原始图片资源。ani.txt为动画的描述信息，内容如下：

; 动画类型,宽，高，循环次数（0为无限循环）

frame,60,60,1

; 帧类型，时长，图片资源，图片区域（可选: (x,y,w,h)，

; 相对上帧位置+(dx,dy,dw,dh) ），输出区域（可选）

; F表示基本图片帧

; F+30表示相同参数的帧重复30次

F,40,0.png,(0,0,60,60),(0,0,60,60)

F+30,40,0.png,+(0,60)

描述文件中以行为单位进行描述。以“；”开头的行为注释，在解析时会自动丢弃。

文件的第一个有效描述行为动画文件基本信息：

frame,60,60,1

frame为动画的类型，表示当前动画为帧动画。

60, 60 为图片的宽和高，单位为象素。

最后的“1”代表了动画循环播放的次数。这是个可选参数，默认值为1。此参数为0时，动画永远循环播放。

动画第一帧的信息由下面内容描述：

F,40,0.png,(0,0,60,60),(0,0,60,60)

其中，F表示帧的类型，现在仅支持“F”。

40代表了此帧显示的时长，单位：ms。

0.png指定了此帧使用的原始图片。

最后的两组坐标分别描述了此帧的源坐标和目的坐标。其格式为(x,y,w,h)。

在本行描述中，它表示从0.png的0,0处选取宽60高60的区域输出到动画显示区域的0,0处，输出图像的宽高为60x60。

源/目的坐标都是可选的，源坐标默认值为(0,0,图片宽,图片高)，目的坐标默认值为(0,0,动画宽,动画高)。

坐标中的宽，高参数也可以省略，省略后将使用其默认值中的宽高。

除了使用绝对坐标外，还可以使用相对坐标。其格式为+(dx,dy,dw,dh)。它表示源/目的坐标使用上一帧的源/目的坐标偏移而来。其默认值为+(0,0,0,0)。

当有多帧动画使用相同的参数时，可以通过如下形式复用帧描述：

F+30,40,0.png,+(0,60)

在此行描述中，F+30表示当前帧描述重复30次。

## 单色图片

在XUL中，可以使用一类特定格式的地址表示一些特殊的“图片”资源。

如下：

@color:COLOR\_VALUE

@color:COLOR\_VALUE,radiusX,radiusY

@color:COLOR\_VALUE,ltRadius,rtRadius,rbRadius,lbRadius

@color:COLOR\_VALUE,ltX,ltY,rtX,rtY,rbX,rbY,lbX,lbY

表示仅由颜色COLOR\_VALUE填充的单色图片。此类图片无需加载，用来替换简单（单色）的png/jpg图片可以极大的减少资源占用。

# 扩展组件

area、item元素除了基本显示行为外，还可以通过type指定其类型，用于扩展元素行为。可用的扩展组件类型如下：

## Label组件

|  |  |
| --- | --- |
| 属性 | 说明 |
| text | 文本值 |
| marquee | 文本自动滚动  参数：  speed，delay，interval，space  mode，speed，delay，interval，space  mode为滚动模式，默认为超长滚动  inverse表示反向滚动  speed为文本滚动速度（单位ms）。单字（中文）滚动的时间  delay启动延迟（单位ms）  interval两次滚动时间间隔（单位ms）  space首尾间距（象素或可视区域百分比） |
| multi-line | true/false，是否为多行文本，默认为false |
| auto-wrap | true/false，文本是否自动换行，仅对多行文本有效，默认为false |
| ellipsis | true/false，文本截断是否显示省略号，默认为false |
| scrollbar | 参考slider组件 |
| scroll-pos-x  scroll-pos-y | 滚动偏移, 设置文本滚动偏移, 仅在元素重新布局时影响文本滚动位置. |

|  |  |
| --- | --- |
| 样式 | 说明 |
| layout-mode | ltr 默认值，从左向右布局  rtl 从右向左布局，自动翻转x对齐方式及滚动方向。 |
|  |  |

## Spanned Label组件

|  |  |
| --- | --- |
| 属性 | 说明 |
| text | 文本值，使用spanned text语法 |
| scrollbar | 参考slider组件 |

## Image组件

Image组件继承自Label组件，具有Label组件所有的属性和功能。文本绘制于最顶层。

|  |  |
| --- | --- |
| 属性 | 说明 |
| img.?  img.?.mode  img.?.align  img.?.width  img.?.height  img.?.visible  img.?.round-rect  img.?.shadow  img.?.auto-hide  img.?.padding  img.?.fade-in  img.?.reuse | 图层图片。  Image可以支持4个图层，img.0、img.1、img.2、img.3。  img.0在最底层，img.3在最上层。  每个图层都可以分别设置以下属性：   1. mode 图层缩放模式。center居中，stretch拉伸（默认） 2. align对齐模式（x-align, y-align）。仅居中有效。 3. width/height图层宽高。只有宽高时，保持比例缩放。图片元素自动大小时（元素的width/height为空或auto），图片元素自动适应最大的图层尺寸；如果有文本内容，且文本尺寸大于所有图层，则使用文本尺寸。   如果图层尺寸为match\_content，则此图层的width或height不参与图片元素的尺寸计算。   1. visible图层是否可见（true/false）。 2. shadow图层投影。水平偏移，竖直偏移，尺寸，颜色 3. round-rect图层圆角，格式：   x-radius，y-radius  left-top, right-top, right-bottom,left-bottom  ltX, ltY, rtX, rtY, rbX, rbY, lbX, lbY   1. auto-hide 自动隐藏模式。below：上层有stretch模式图片时隐藏 2. padding 图片边距，（top，left，right，bottom），负值会使图片放大。 3. fade-in 图片淡入效果，当图片加载成功时淡入。默认值为false。也可以指定动画时长（单位：ms）如：500，默认为300。 4. reuse图片重用，默认为false。当设置为true时，图片在回收image图层时进入重用缓存。当有大量不同内容的同尺寸图片时（如图片列表）可用此功能减少图片的重新创建。 |
| max-layers | 指定最大图层数量，默认为4，最大值为8 |
|  |  |

|  |  |
| --- | --- |
| 样式 | 说明 |
| do-not-match-text | 元素match\_content时不考虑text的尺寸  布尔类型，默认为false  格式：宽/高 *或* 宽，高 |
| layout-mode | ltr 默认值，从左向右布局  rtl 从右向左布局，自动翻转x对齐方式及左/右padding。 |
|  |  |

## Slider容器

|  |  |
| --- | --- |
| 属性 | 说明 |
| direction | 方向。水平（horizontal）/竖直（vertical）  反向水平/竖直布局：reverse-horizontal/reverse-vertical  ***反向布局仅做简单布局用，子元素不应该动态改变布局（尺寸），否则可能会引起布局错乱。*** |
| auto-scroll | 自动跟随焦点滚动。enabled/disabled。默认为enabled |
| align | 子元素在X,Y方向的对齐方式。如： 0.5,0.5表示居中对齐 |
| scrollbar | 设置滚动条样式，默认为none，无滚动条  简单样式：  simple, [基本参数,[背景参数]]  基本参数如下：  位置(0~1),隐藏延迟(ms，-1为不隐藏),宽度,圆角尺寸(不能大于宽度/2),颜色,阴影尺寸,阴影颜色  背景参数如下：  背景色  背景色,边距  背景色,滚动范围色,边距  背景色,滚动范围色,背景圆角,边距  背景色,边距上,左,右,下  背景色,滚动范围色,边距上,左,右,下  背景色,滚动范围色,背景圆角,边距上,左,右,下  例(simple+基本参数)：  simple,1.0,3000,4.0,2.0,80FFFFFF,2.0,FF000000 |
| indicator | 是否使用指示器。enabled/disabled  当slider在某一方向上可以滚动时，显示相应方向的指示器图片。 |
| indicator.left  indicator.up  indicator.right  indicator.down | width, height, x-align, y-align, image-url  width, height, image-url  image-url |
| loop | 循环模式。默认为false。当元素大于2个及长度/宽度超出可见范围时，自动循环滚动所有元素。 |
| lock-focus | 焦点锁定模式。参数：alignment [,align-point]  此样式使slider在必要时，自动对焦点元素调用makeChildVisible方法，将焦点元素固定在特定位置。  当alignment为-1时，表示自动计算alignment。此时，元素对齐位置为其在整个列表中位置的百分比。  焦点锁定时默认是以当前的焦点元素做为锁定目标, 如果焦点元素的父级元素有lock-focus-target样式, 则以距slider最近的一级具有lock-focus-target样式的元素做为锁定目标. |
| lock-dynamic-focus | true/false. 默认为false  当slider中有dynamic focus元素时, 保持此元素在可见区域, 配合lock-focus使用 |

|  |  |
| --- | --- |
| 样式 | 说明 |
| clip-children  clip-focus | 绘制时是否裁减子级元素（true/false） 默认为true  绘制时是否裁减焦点元素（true/false） 默认为true |
|  |  |

## Page Slider容器

|  |  |
| --- | --- |
| 属性 | 说明 |
| direction | 方向。水平（horizontal）/竖直（vertical） |
| indicator | 是否显示指示器，enabled/disabled。默认为disabled |
| indicator.style | 参数格式：  对齐参数(0~1)，默认样式  对齐参数(0~1)，当前页样式，默认样式  可用的样式有下三种：  短线：dash, 宽，高，边距，颜色  点：dot, 直径，边距，颜色  图片：image，宽，高，边距，地址  示例：  0, dash,16,4,0,FF00FF00,dot,6,0,FF008000 |
| indicator.gap | 所有indicator元素的基本间隔 |
| indicator.align | indicator区域对齐方式（x-align, y-align）。范围0~1之间。 |
| loop | 是否循环切换页面。默认为true |
| image-gc | 页面切换自动回收图片，auto/disabled。默认为auto |
| animation-speed | 动画速度，-1为无动画 |
| animation-type | 动画效果，slide为滑动，flip为翻转。默认为slide |
| switching-mode | 页面切换模式，  none，不切换（仅由代码控制切换）  focus，跟随焦点切换（默认）  auto，自动切换（不支持） |
| preload-page | 进行预加载的页面数量. 其值表示从当前页开始, 向其前后各加载多少个页面的数据. 默认为1, 表示前后各预加载1页数据. 当为disable或0时表示不进行预加载. |

|  |  |
| --- | --- |
| 样式 | 说明 |
| clip-children  clip-focus | 绘制时是否裁减子级元素（true/false） 默认为true  绘制时是否裁减焦点元素（true/false） 默认为false |
|  |  |

## Grid容器

|  |  |
| --- | --- |
| 属性 | 说明 |
| direction | 方向。水平（horizontal）/竖直（vertical） |
|  |  |

## Group/Radio/Check容器

多选/单选容器。当分组子元素被click时，自动切换选中状态。

|  |  |
| --- | --- |
| 属性 | 说明 |
| checked-class | 选中元素的识别class名称。  具有此class的元素被认为是选中状态。元素选中后会自动添加此class，取消选中时会自动删除此class。  默认值为group-checked。 |
| group | 用于指定分组(一个group组件可以定义多个不同的子分组)，为分组的xml描述信息。  如无分组信息，则默认所有子元素为分组成员。工作模式由Group元素类型(type=check或radio)决定。Group默认为多选模式(check)。  为一条或多条如下分组描述信息：  <group>mode,group-class,checked-class</group>  mode表示分组的类型，目前支持多选（check）及单选（radio）模式。  group-class表示分组的识别class名，具有此class的元素才被认为是分组的成员。  checked-class为选中元素的识别class名称。具有此class的元素被认为是选中状态。元素选中后会自动添加此class，取消选中时会自动删除此class。 |
|  |  |

## Custom组件

|  |  |
| --- | --- |
| 属性 | 说明 |
|  |  |

## Layer容器

图层容器。可以给子元素的绘制添加各种变换效果。

**注意：**当有子元素的id为@layer-mask时，此元素不用做内容显示，而是做为图层的掩码。

|  |  |
| --- | --- |
| 样式 | 说明 |
| opacity | 不透明度。范围0.0 ~1.0 |
| translate | 平移变换。参数 dXY或dX, dY |
| translate-x | 平移变换x |
| translate-y | 平移变换y |
| rotate | 旋转变换。参数格式：  X,Y 或  X,Y,Z 或  X,Y, bX, bY 或  X,Y, Z, bX, bY,bZ  其中，X,Y,Z(逆时针)为旋转角度；bX, bY, bZ为旋转中心偏移，bX, bY为百分比，bZ为像素。 |
| rotate-x | 旋转变换x。参数格式：X 或 X, bX |
| rotate-y | 旋转变换y。参数格式：Y 或 Y, bY |
| rotate-z | 旋转变换z。参数格式：Z 或 Z, bZ |
| rotate-center | 旋转变换中心。参数格式：bX, bY 或 bX, bY, bZ |
| rotate-center-x | 旋转变换中心bX |
| rotate-center-y | 旋转变换中心bY |
| rotate-center-z | 旋转变换中心bZ |
| lighting-color-filter | 颜色过滤。mulColor 或 mulColor, addColor |
| round-rect | 圆角。格式  WH  W, H  lt, rt, rb, lb  ltW, ltH, rtW, rbH, rbW, rbH, lbW, lbH |
|  |  |

## Component容器

通过area元素实例化component组件。此容器内不应该添加其它子元素，否则在实例化时会被自动删除。

|  |  |
| --- | --- |
| 属性 | 说明 |
| component | 引用的组件id |
|  |  |

## Massive 容器

Massive容器用于显示大规模数据项。数据项通过binding指定，或由用户代码添加。**注意：当使用binding绑定数据时只需绑定数据项的父级元素，避免选择大量子元素引起性能下降**。

数据显示的样式通过id为“@item-template”的template指定。此template不要设置binding属性。且模板中的布局不能使用template元素做为根级元素。

如下：

<area type="massive" binding="#massive-data/">  
 <attr name="width">790</attr>  
 <attr name="height">auto</attr>  
 <attr name="arrangement">grid</attr>  
 <attr name="direction">horizontal</attr>  
 <style name="border">2,FFFF0000</style>  
 <template id="@item-template">  
 <item type="image" class="button-image">  
 <attr name="text" binding="name"/>  
 </item>  
 </template>  
</area>

|  |  |
| --- | --- |
| 属性 | 说明 |
| arrangement | 元素布局模式：  linear 线性布局  grid 网格布局 （默认）  linear-fixed 固定尺寸线性布局  grid-fixed 固定尺寸网格布局  *固定尺寸布局要求所有子元素大小相同，便于布局计算* |
| direction | 布局方向：vertical 纵向、 horizontal 水平 |
| minimum-item | 最小加载元素数量：默认为16  当数据项足够时，至少会预先创建minimum-item个元素。此值过小会导致元素显示不全；过多会影响性能。建议值为2页元素的个数。 |
| cache-pages | 最小缓存数据页数，默认为前后0.5页 |
|  |  |

# XUL脚本

XUL中可以通过动作（Action）绑定脚本，直接操作UI元素。XUL从设计上，可以同时支持不同的脚本语言（目前仅支持JavaScript）。

## 概念

由于XUL中，脚本只能绑定在元素的动作中。所以，脚本在执行时其默认的环境上下文为动作绑定的目标元素。这点尤为重要！！！

以JavaScript为例，脚本被触发时，其运行环境（scope）和当前上下文（this）都指向事件源对象。在此环境下定义的变量，函数都属于事件源对象。在其它上下文中将无法使用。

如果要定义全局对象，可以通过为全局上下文对象--XUL添加成员实现。

## 对象

### XUL对象

此对象为脚本运行的根级环境上下文。

### logger对象

此对象为全局对象，用于输出调试信息。

#### logger.d(messages)

输出调试信息

#### logger.w(messages)

输出警告信息

#### logger.e(messages)

输出错误信息

### View对象

View对象为所有UI元素的基类。

#### View.getStyle(name)

读取元素的原始style。

#### View.setStyle(name, style)

设置元素的原始style。

#### View.getAttr(name)

读取元素的原始attr。

#### View.setAttr(name, attr)

设置元素的原始attr。

#### View.getAction(name)

读取元素的原始action数据。

#### View.getData(name)

读取元素的原始data数据。

#### View.viewWidth

得到元素的可见宽度。只读属性。

#### View.viewHeight

得到元素的可见高度。只读属性。

#### View.viewX

得到元素的可见X坐标。相对于屏幕。只读属性。

#### View.viewY

得到元素的可见Y坐标。相对于屏幕。只读属性。

#### View.text

读/写元素的文本属性，与setAttr(“text”,value)/getAttr(“text”)等价。

#### View.id

读取元素id。只读属性。

#### View.type

读取元素类型。只读属性。

#### View.binding

读取元素数据绑定选择器。只读属性。

#### View.parent

得到元素的直接父元素。只读属性。

#### View.findParentById(id)

根据id查找父元素，返回第一项符合条件的元素。

#### View.findParentByClass(cls)

根据class查找父元素，返回第一项符合条件的元素。

#### View.findParentByType(type)

根据type查找父元素，返回第一项符合条件的元素。

#### View.ownerPage

得到元素所属的页面。只读属性。

#### View.isChildOf(view)

检测元素是否为另一元素的子级。

#### View.isFocusable

读取元素是否可设置焦点。只读属性。

#### View.isVisible

读取元素是否可见。只读属性。

#### View.isEnabled

读/写元素可用状态。

#### View.isFocused

读取元素焦点状态。只读属性。

#### View.hasFocus

读取元素及其子元素焦点状态。只读属性。

#### View.requestFocus()

将元素设置为焦点。

#### View.killFocus()

清除元素焦点。

#### View.classList

读取元素的类列表，修改返回的数组不影响元素本身。

#### View.hasClass(clsName)

判断元素是否具有相应的class

#### View.addClass(clsName)

给元素添加新的class

#### View.removeClass(clsName)

删除元素的class

#### View.isBindingSuccess

读取元素是否绑定数据成功。只读属性。

#### View.isBindingReady

读取元素是否绑定数据完成。只读属性。

#### View.dispatchEvent(“event”, “type”, “command”)

发送事件给宿主应用。

#### View.fireEvent(“event”)

触发指定的事件。注意：触发脚本绑定的事件可能引起递归调用！！！

#### View.blinkClass(“className0”,…)

给元素添加样式，触发动画效果，并在动画结束后删除所有样式。

### Area对象

继承自View对象，为所有容器的基类

#### Area.findItemById(id)

根据id查找子元素，返回第一项符合条件的元素。

#### Area.findItemsByClass(clsName)

根据类名查找子元素，返回符合条件的元素数组。

#### Area.children

容器中所有元素的集合。为一个数组对象。修改数组不改变容器的子元素。

#### Area. setDynamicFocus(view)

设置容器的动态焦点。

### Slider对象

继承自Area对象

#### Slider.scrollTo(pos, [animation] )

设置slider的滚动位置

#### Slider.scrollRange

得到当前slider的最大滚动范围

#### Slider.scrollPos

得到slider滚动目标位置

#### Slider.scrollDelta

得到slider滚动偏移。即slider目标位置与当前滚动位置的差值。

当slider有滚动动画时，scrollPos得到的是滚动结束时的位置；而当前内容显示的偏移与scrollPos可能存在一定的差异。scrollPos+scrollDelta即为动画过程中的内当前容显示偏移。

#### Slider.activateScrollBar()

激活滚动条。

#### Slider.makeChildVisible( child, [animation] )

强制将子元素滚动到可见区域

#### Slider.makeChildVisible( child, align, [animation] )

强制将子元素滚动到可见区域中由align指定的位置。

align为对齐位置。取值范围为0~1.0。

算法：**对齐位置 = (sliderSize – childSize)\*align**

#### Slider.makeChildVisible( child, align,alignPoint, [animation] )

alignPoint为子元素的对齐点。

算法：**对齐位置 = sliderSize\*align - childSize\*alignPoint**

#### Slider.scrollByPage( pages, [animation] )

按页（可见区域尺寸）滚动内容。

pages为正向右/下滚动，为负向左/上滚动。

### PageSlider对象

继承自Area对象

#### PageSlider.pageCount

当前PageSlider中可见页面数量。只读属性。

#### PageSlider.currentPage

当前页面id。只读属性。

#### PageSlider.setCurrentPage(idx)

设置当前显示页面，索引idx从0开始。

#### PageSlider.syncPages()

同步PageSlider对象的子页面。当PageSlider显示前，它会自动同步页面内容。但在页面未显示时，其子页面内容可能为空、pageCount为0，设置当前页面没有效果；此时，可以通过syncPages同步强制同步页面数据。

#### PageSlider.slideNext()

内容向后翻页。PageSlider为纵向时，向下翻页；为横向时，向右翻页。

#### PageSlider.slidePrev()

内容向前翻页。PageSlider为纵向时，向上翻页；为横向时，向左翻页。

#### PageSlider.slideLeft()

内容向左翻页。

#### PageSlider.slideRight()

内容向右翻页。

#### PageSlider.slideUp()

内容向上翻页。

#### PageSlider.slideDown()

内容向下翻页。

### Group对象

### Page对象

继承自View对象

#### Page.findItemById(id)

根据id查找元素，返回第一项id符合的元素。

#### Page.findItemsByClass(clsName)

根据class查找元素，返回符合条件的元素数组。

#### Page.currentFocus

得到当前焦点元素。无焦点时返回空。只读属性。

#### Page.queryBindingDataString(dataQueryCommand)

查询当前页面的binding数据。返回结果字符串。查询语法请参考[数据选择器](#_数据选择器)。

#### Page.refreshBinding(bindingId, [bindingURL ])

刷新binding数据。

#### Page.pushStates ( XulView [cmd, [cmd, […] ]], [XulView … ] )

更改一系列的子元素的属性/样式/类/焦点/可用状态，并将旧状态记录在状态栈中。当调用popStates或按“返回”键时，恢复所有元素状态。可实现页面状态的历史记录功能。

可用的命令有：

|  |  |  |
| --- | --- | --- |
| 命令 | 参数 | 说明 |
| setEnabled |  | 设置元素为可用状态 |
| setDisabled |  | 设置元素为不可用状态 |
| requestFocus |  | 设置元素为焦点元素 |
| killFocus |  | 设置元素为非焦点状态 |
| addClass | className | 添加新的类 |
| removeClass | className | 删除已有的类 |
| setStyle | styleName, styleValue | 设置内联样式 |
| removeStyle | styleName | 删除内联样式 |
| setAttr | attrName, attrValue | 设置内联属性 |
| removeAttr | attrName | 删除内联属性 |
|  |  |  |

如：this.ownerPage.pushStates( view1, “addClass”, “new-class”, “removeClass”, “old-class”, “addClass”, “new-class2”, view2, “setAttr”, “text”, “text-value”, “removeStyle”, “display”);

此调用将给view1添加两个新的class “new-class”及“new-class2”，并删除旧class “old-class”；给view2更改text属性为“text-value”，删除display样式。在弹出状态时，所有更改将被还原成旧值。

#### Page.popStates([discard = false])

恢复状态栈中保存的元素状态，当没有可恢复状态时，返回false。

当discard参数为true时，仅把状态弹出，不恢复元素旧状态值。

#### Page.popAllStates([discard = false])

恢复状态栈中保存的所有元素状态。

当discard参数为true时，仅把状态弹出，不恢复元素旧状态值。

### Massive对象

#### Massive.getItemIdx(item)

得到子元素在massive容器中的索引号

#### Massive.removeItem(item)

删除容器中的元素

#### Massive.makeChildVisible(ownerSlider, itemIdx [, animation = true])

#### Massive.makeChildVisible(ownerSlider, itemIdx, align [, animation = true])

#### Massive.makeChildVisible(ownerSlider, itemIdx, align, alignPoint [, animation = true])

将massive容器的子元素移动到可见区域。注意，只有子元素为固定尺寸（isFixed）时，才能保证正确的将不可见元素移动到可见区域。如果子元素尺寸不固定，调用此方法可能导致未知结果。

ownerSlider为massive容器的父级slider，用于控制massive容器的可见范围。如果为null，则自动查找massive的最近父级slider。

itemIdx为要移到可见区域的子元素索引。

align, alignPoint请参考slider的makeChildVisible方法

animation 为动画开关，默认为true

#### Massive.syncContentView(item)

同步massive容器的数据/视图

#### Massive.clear()

删除所有子元素

#### Massive.isFixed

子元素是否为固定尺寸

#### Massive.itemNum

子元素的数量

# 应用实例