Htmlayout

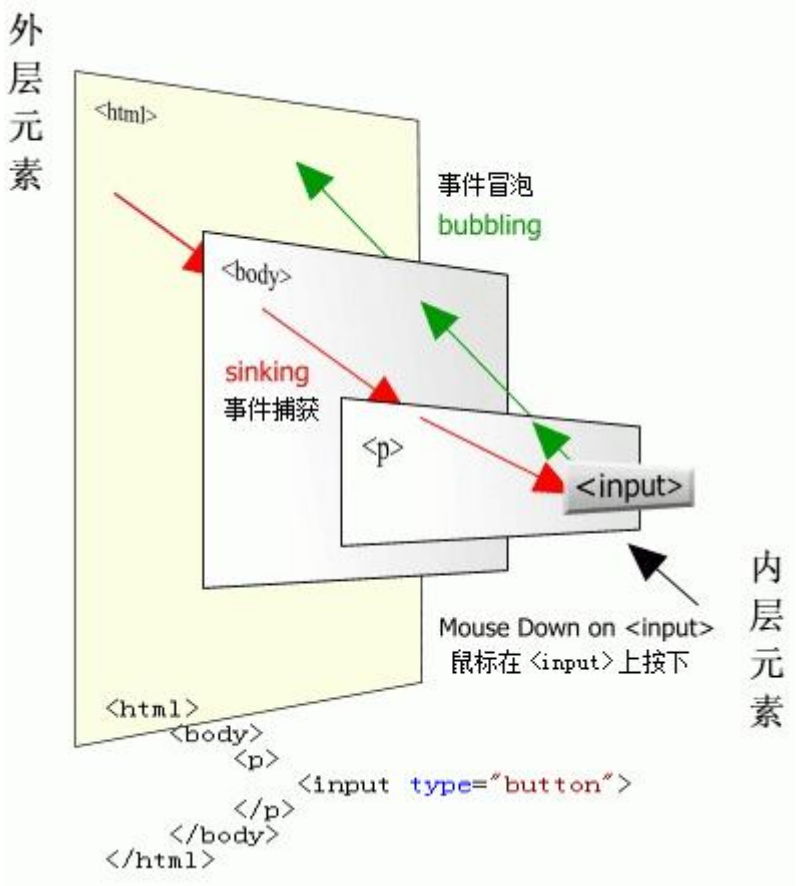
手册

此手册由 aardio 论坛用户 veluxa 整理并编写

目录

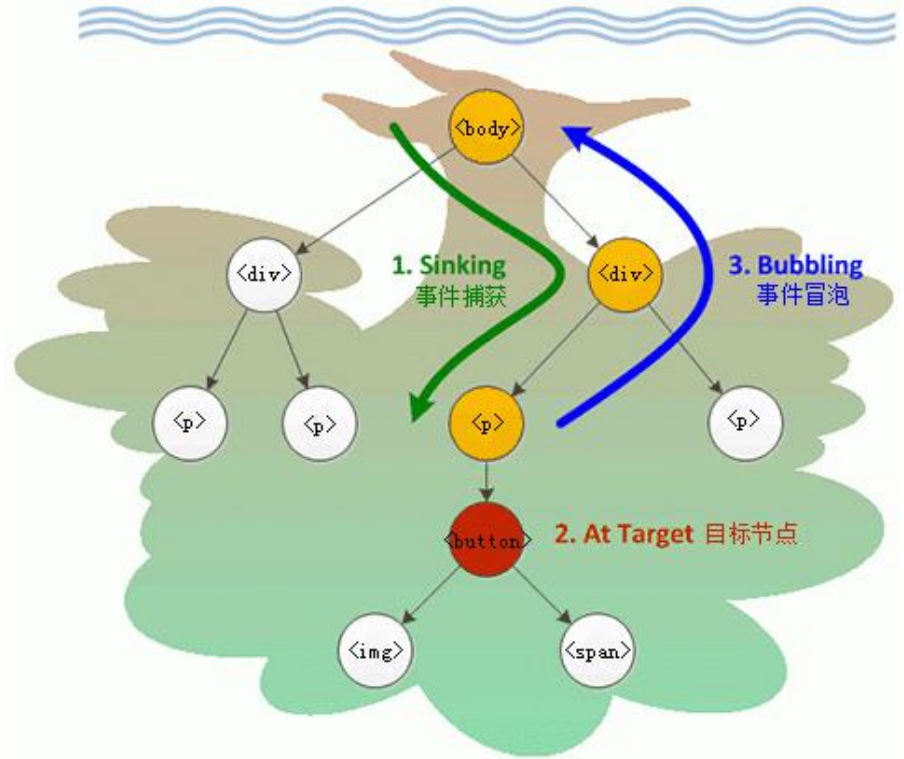
一、事件捕获与冒泡过程

在 HTML 浏览器中，一般点击元素等交互事件会首先由外层节点向内层目标节点传递 - 这是捕获（sinking）过程，然后再返过来由内层目标节点向外层节点传递 - 这是冒泡（ bubbling ）过程



这里的捕获、冒泡是相对于 HTML 源码里元素位置的概念。

例如 上面的 input 元素在屏幕上他是显示在最前面，而在 HTML 源码中，他是在最里面，HTML 根节点才是在最外面



所有浏览器引擎都支持事件冒泡，IE11 以下不支持事件捕获（ 也就是不支持 W3C 标准的函数 addEventListener ）

默认情况下，事件使用冒泡事件流(所有浏览器引擎都支持)，不使用捕获事件流。

HTMLayout 也一样，默认所有的事件都是在冒泡过程中处理，包括所有内置的默认事件。
例如点击一个超链接，会在冒泡过程中默认触发内置的 onHyperlinkClick 事件以打开超链接，如果希望自定义这个事件，调用其他浏览器打开一个超链接并且阻止默认的事件，那么就要在捕获过程中拦截 onHyperlinkClick 事件（ 因为 捕获过程 早于 冒泡传递过程).

在 aardio 中，将事件写在 sinking 表中就是在 捕获过程中触发，示例：

```
import process;
wbLayout.sinking = {
    onHyperlinkClick = function(ltTarget,ltOwner,reason,behaviorParams) {
        process.execute( ltTarget.href );
        return true;//阻止事件流继续传递
    }
}

wbLayout.html = /**
<!doctype html>
<html>
<head>
    <style type="text/css">
    html,body{ height:100%; margin:0; }
    </style>
</head>
<body>
     <a href="http://bbs.aardio.com">http://bbs.aardio.com</a>
</body>
</html>
**/
```

再举个例子：
文本框得到焦点（onFocusGot）、失去焦点（ onFocusLost ） 也是被默认的 behavior 处理并阻止传递了，
所以也需要在捕获过程中处理，主要代码如下：

```
wbLayout.sinking = {
    onFocusGot = function(ltTarget,ltOwner,focusParams) {}
    onFocusLost = function(ltTarget,ltOwner,focusParams) {}
}
```

sinking.onFormSubmit 阻止表单默认提交

```
wbLayout.sinking = {
    onFormSubmit = function(ltTarget,ltOwner,reason,behaviorParams) {
        io.print(ltOwner.getElementById("login").value );
        return true;
    }
}
```

sinking.onKeyDown 处理文本框左右方向键

```
wbLayout.sinking = {
    onKeyDown = function (ltTarget,ltEle,keyCode, altState, ltKeyParams) {
        if( keyCode == 0x25/*_VK_LEFT*/ ){

        }
    }
}
```

二、CSS 属性

1. Htmlyout 独有的特性

- 渐变背景，见 background-color 属性的定义。
- 扩展的图像填充，见 background-position 和 background-repeat 属性的定义。
- %% 长度单位 - "占自由空间的百分比"。
- flow 属性。
- 前景(Foreground)图像。
- hit-margin 属性。
- @include "mime-type" url(...) [媒体类型列表]; - CSS 中的内嵌脚本。
  示例: <style> @include "text/tiscript" url(script.tis) screen; </style>
  等价于 <script type="text/tiscript" src="script.tis" />(除了媒体类型条件)。

支持所有的 CSS1 基本概念和层叠顺序，也支持 CSS 2.1 的属性选择器和一些基本的 CSS3 选择器。

2. 自定义 CSS 属性

支持自定义 CSS 属性。所有的自定义属性名称以'-'字符开头。自定义属性不会从父元素继承到子元素。 下面是一个自定义属性的声明:

p { border: 1px solid red; -custom: "value"; }

在上面的这个声明后，所有的<p>元素都会有值为"value"的-custom 样式属性。

自定义属性的值可以为下面的值之一:

- 字符串 (使用""括起来), 示例: "string" ;
- nmtoken(Name Token 的缩写，中文意思就是名称记号，它表示由一个或多个字母、数字、句点、连字号或底线所组成的一个名称), 示例: some-token ;
- 数字值, 示例: 3.1415 ;
- url, 示例: url(http://something.com);
- CSS 选择器, 示例: selector(#id > .cls);

宿主程序可以使用自定义属性来实现任何目的。 引擎内部不会翻译解释这些属性。

示例:

```
import win.ui;
/*DSG{{*/
winform = ..win.form( text="aardio Form";bottom=399;parent=...;right=599;border="resizable" )
```

```
winform.add(
layoutWindow={ dr=1;dl=1;notify=1;right=580;left=10;dt=1;top=12;z=1;db=1;bottom=382;multiline=1;cls="edit" }
)
/*}}*/

import web.layout;
wbLayout = web.layout(winform.layoutWindow);

//自定义一个名字为 command 的 behavior
namespace web.layout.behavior.command{
    onMouseDown = function (ltTarget,ltEle, x,y,mouseParams) {

        var message = ltEle.getCustomAttribute("message"); //在 CSS 中定义该属性时前面加一个横线
        var 属性 = ltEle.getCustomAttribute("属性"); //这个函数也可以用来读取普通 HTML 属性,也可以直接写
        var 属性 = ltEle.属性; //这样写仅支持 HTML 属性,不支持 CSS 自定义属性

        ..io.open()
        ..io.print("鼠标单击了,节点 message = ",message)

        return true;//所有事件返回 true 终止事件.
    }
}

wbLayout.html =/***
  <div id="my-button" 属性="值">请点击这里</div>
***/

wbLayout.css = /**
  #my-button{
    behavior:command; //绑定 behavior
    -message:"在 CSS 中以横线开头表示添加自定义属性,也可以直接写在 HTML 中";
  }
**/

winform.show()
win.loopMessage();
```

3. 标准 CSS 属性（支持的 CSS 属性列表）

3.1 字体和段落属性

font	font 是一个简写属性，在一个声明中设置所有字体属性，用于一次设置元素字体的两个或更多方面。 注释：此属性也有第六个值："line-height"，可设置行间距。 可以按顺序设置如下属性： font-style font-variant font-weight

	font-size/line-height font-family 在引擎中， font 也支持下面的系统字体: system 输入元素默认的系统 UI 字体 system-menu 系统菜单字体 system-caption 窗口标题字体 system-status 状态栏和气泡提示系统字体 注意: 上面的这些字体名称只有在 font 样式属性中才有效 - 在 font-family 属性中，这些名称会被忽略。之所以这样做是因为这样比较简单。比如拿 font:system-menu;来说，它是 font-family、font-size、font-weight、font-variant 四个属性的简写形式。
font-family	指定下面中的值之一: family-name 系统中安装的任何可用的字体家族。例如：Times、Helvetica、Zapf-Chancery、Western、Courier 等 generic-name 后面的这些字体家族: serif、sans-serif、cursive、fantasy 或 monospace
font-size	指定下面中的值之一: absolute-size 一套关键字集，它们指定字体的预定义大小。命名的字体尺寸根据用户的字体设置偏好来缩放尺寸。可能的值如下: xx-small、x-small、small、medium、large、x-large、xx-large relative-size 一套关键字集，它们指定的尺寸是相对于父对象字体尺寸。可能的值如下: larger、smaller length 浮点数字, 后面跟个绝对长度单位(cm、mm、in、pt、pc、px) 或一个相对长度单位(em、ex)。要获取更多关于长度单位的信息，请参见 CSS 长度单位指南 percentage 整数值, 后面跟个百分号(%)。相对于父对象字体尺寸的百分比
font-style	指定下面中的值之一: normal 默认. 字体是正常的 italic 字体是斜体的 oblique 字体是斜体的
font-weight	指定下面中的值之一: normal 默认字体 bold 字体是粗体的 bolder 字体比常规的粗体更粗 lighter 比正常的字体更细些 100 字体最起码会比 200 更细 200 字体最起码比 100 粗，而比 300 细 300 字体最起码比 200 粗，而比 400 细 400 字体为正常字体 500 字体最起码比 400 粗，而比 600 细 600 字体最起码比 500 粗，而比 700 细 700 字体为粗体 800 字体最起码比 700 粗，而比 900 细 900 字体最起码比 800 粗
letter-spacing	不支持
line-height	指定下面中的值之一: normal 默认高度 height 浮点数字, 后面跟个绝对长度单位(cm、mm、in、pt、pc、px) 或一个相对长度单位(e

	m、ex) percentage 整数值，后面跟个百分号(%)。相对于父对象高度的百分比 calc(...) calc 函数
text-align	指定下面中的值之一: left 默认，文本左对齐 right 文本右对齐 center 文本居中对齐 justify 文本自动调整
text-decoration	指定下面中的值之一: none 默认。文本不做装饰 underline 文本有下划线 overline 文本有上划线 line-through 文本有删除线 blink 没有实现
text-indent	指定首行文本的缩进。 length 浮点数字，后面跟个绝对长度单位(cm、mm、in、pt、pc、px) 或一个相对长度单位(em、ex) percentage 整数值，后面跟个百分号(%)。相对于父对象宽度的百分比
text-overflow	指定下面中的值之一，用于决定超出布局区域的文本是否省略(...)显示。 ellipsis 超出的文本显示省略号(...) clip 默认。简单的裁剪内容区，超出的文本不显示
text-transform	琐碎的文本变换，可接受以下值: none | uppercase | lowercase | capitalize 默认值: none.
vertical-align	垂直对齐方式，指定下面中的值之一: baseline 默认。将支持**垂直对齐**的对象内容对齐到基线上 sub 将文本对齐到支持**垂直对齐**对象的下标位置 super 将文本对齐到支持**垂直对齐**对象的上标位置 top 将内容区对齐到支持**垂直对齐**的对象的顶部 middle 将内容区对齐到支持**垂直对齐**的对象的居中位置 bottom 将内容区对齐到支持**垂直对齐**的对象的底部 text-top 将文本对齐到支持**垂直对齐**的对象的顶部 text-bottom 将文本对齐到支持**垂直对齐**的对象的底部
white-space	指定下面中的值之一: normal 行可以在允许的中断点换行，中断点取决于实际的断行规则 none 行不可以中断，超出块盒的文本将不会自动适应 pre 文本中的换行和其他空白符保留 prewrap 文本中的换行和其他空白符保留，同时内容允许在需要换行的地方换行。这与<textarea>的wrap="on"时的模式相同。非标准属性值
text-wrap	指定下面中的值之一: normal 只在允许的换行点进行换行，如空格，TAB 和回车 none 不换行，元素无法容纳的文本会溢出 unrestricted 在任意两个字符间换行
text-selection-color	指定被选择文本的颜色(输入元素和有 behavior:htmlarea 行为的元素)。非标准。继承属性
text-selection-background-color	指定被选择文本的背景颜色(输入元素和有 behavior:htmlarea 行为的元素)。非标准。继承属性
text-selection	<text-color> <background-text-color>

	指定被选择文本的颜色(包括背景和前景)。是上面两个属性的简写形式。非标准属性。

3.2 颜色和背景属性

background	background 是一个简写属性，在一个声明中设置所有的背景属性。 可以设置如下属性： background-color background-position background-size background-repeat background-origin background-clip background-attachment background-image
background-attachment	指定下面中的值之一： scroll 默认，背景图像随着文档的滚动而滚动 fixed 背景图像固定在可视区域，不随着文档的滚动而滚动
background-color	指定下面中的值之一： transparent 默认，背景是透明，可以看到父对象的颜色 color 任何一个颜色值，包括颜色表中的颜色值。 color-left-top, 渐变填充。背景渐变填充的四个角的颜色。非标准。 color-right-top, **HTMLayout 独有属性** color-right-bottom, color-left-bottom
background-image	指定下面中的值之一： none 默认。背景是透明的，可见父对象的颜色 url(sUrl) 背景图像的地址，其中，sUrl 可以是绝对或相对 URL
background-position	指定下面中的值之一： length 浮点数字，后面跟个绝对长度单位(cm、mm、in、pt、pc、px)或一个相对长度单位(em、ex) percentage 整数值，后面跟个百分号(%)。相对于父对象宽度或高度的百分比 top 垂直对齐值。可能有以下值 ： center top 垂直对齐到顶部 bottom center 垂直居中 bottom 垂直对齐到底部 left 水平对齐值。可以有以下值： center left 水平对齐到左边 right center 水平居中 right 水平对齐到右边 left-side-length **可扩展填充(九宫格)**图像外边距 top-side-length 参见 HtmLayout 文档中 **可扩展平铺**节的内容 right-side-length 要使用可扩展填充背景，需要设置 **background-repeat** 属性的 bottm-side-length 值为'**expand**' 非标准。HTMLayout 独有 calc(...) calc 函数 .style1 { background-position:top center } .style2 { background-position:0 0 }

background-position-left background-position-right background-position-top background-position-bottom	用于 background-repeat=expand 模式，值可以为：stretch、no-repeat，更多参见 HTML ayout 界面贴图技术
background-repeat	指定下面中的值之一： repeat 默认。图像在水平和垂直方向上平铺 no-repeat 图像不重复(即不平铺) repeat-x 图像在水平方向上平铺 repeat-y 图像在垂直方向上平铺 expand **可扩展填充(九宫格)** 模式 stretch[keep-ratio] 图像被拉伸以铺满整个背景，与<IMG>元素中的图像渲染方式相同。如果指定了 keep-ratio，则拉伸时会保持图像的宽高比例。要设置背景图像的位置，请使用 background-position 属性
color	设置对象中文本的颜色。

3.3 前景图像属性

前景(Foreground)图像是 HTMLayout 独有的特性。
前景图像拥有于背景图像相同的属性集合。
HTMLayout 中块元素的绘制顺序:

1. 背景(background)
2. 边框(borders)
3. 内容(content)
4. 轮廓(outline)
5. 前景(foreground)

foreground	指定最多 4 个下面的值，值间以空格符分隔，可以任意顺序： <foreground-image url> || <foreground-repeat> || <foreground-attachment> || <foreground-position> || <foreground-color> || <foreground-gradient>
foreground-attachment	指定下面中的值之一： scroll 默认。前景图像随着文档的滚动而滚动 fixed 前景图像固定在可视区域，不随着文档的滚动而滚动
foreground-image	可指定以下值： none 默认。前景是透明的，可见父对象的颜色 url(sUrl) 前景图像的地址，其中，sUrl 可以是绝对或相对 URL
foreground-position	指定下面中的值之一： length 浮点数字，后面跟个绝对长度单位(cm、mm、in、pt、pc、px) 或一个相对长度单位(em、ex) percentage 整数值，后面跟个百分号(%)。相对于父对象宽度或高度的百分比 top 垂直对齐值。可能有以下值 ： center top 垂直对齐到顶部 bottom center 垂直居中 bottom 垂直对齐到底部 left 水平对齐值。可以有以下值： center left 水平对齐到左边 right center 水平居中

	right 水平对齐到右边 left-side-length **可扩展填充(九宫格)**图像外边距。 top-side-length 参见 HtmLayout 文档中 **可扩展平铺**节的内容。 right-side-length 要使用可扩展填充背景，需要设置 **foreground-repeat** 属性的值为' **expand**'。 bottm-side-lengt 非标准。HTMLayout 独有。 calc(...) calc 函数 .style1 { background-position:top center } .style2 { background-position:0 0 }
foreground-position-left foreground-position-right foreground-position-top foreground-position-bottom	用于 background-repeat=expand 模式，值可以为：stretch、no-repeat，更多参见 HTMLayout 界面贴图技术
foreground-repeat	指定下面中的值之一： repeat 默认。图像在水平和垂直方向上平铺 no-repeat 图像不重复(即不平铺) repeat-x 图像在水平方向上平铺 repeat-y 图像在垂直方向上平铺 expand **可扩展填充(九宫格)** 模式 stretch[keep-ratio] 图像被拉伸以铺满整个背景，与<IMG>元素中的图像渲染方式相同。如果指定了 keep-ratio，则拉伸时会保持图像的宽高比例。要设置背景图像的位置，请使用 background-position 属性
foreground-image-cursor	当 foreground-repeat 值为 no-repeat 时，鼠标悬停在前景图片区域时的鼠标指针样式值可以为: auto | crosshair | default | pointer | move | e-resize | ne-resize | nw-resize | n-resize | se-resize | sw-resize | s-resize | w-resize | text | wait | help | progress 或者.cur/.ani 文件类型的 url

3.4 布局属性

border	指定一个或多个下面的值: width, style, color
border-bottom	指定一个或多个下面的值: width, style, color
border-bottom-color	指定颜色表中的颜色名称或 RGB 值。
border-bottom-style	指定下面中的值之一： none 默认。不绘制边框，忽略任何边框宽度 dotted 边框是一条(dotted)虚线 dashed 边框是一条(dashed)虚线 solid 边框是一条(solid)实线 double 边框是画在对象背景上的双线。两条线和它们中间的空间之和等于 **borderWidth**。边框宽度最少得 3 像素宽才能绘制一条双线边框 groove 3D 凹槽边框。其效果取决于 border-color 的值 ridge 3D 垄状边框。其效果取决于 border-color 的值 inset 3D inset 边框。其效果取决于 border-color 的值 outset 3D outset 边框。其效果取决于 border-color 的值
border-bottom-width	指定下面中的值之一：

	medium 默认 thin 比默认宽度窄些 thick 比默认宽度粗些 width 浮点数字, 后面跟个绝对长度单位(cm、mm、in、pt、pc、px) 或一个相对长度单位(em、ex) %% 单位 占剩余空间的百分比 calc(...) calc 函数

border-radius	border-radius 属性是一个简写属性，用于设置四个 border-*-radius 属性。
	提示：该属性允许您为元素添加圆角边框！
	border-radius: none //无圆角
	border-radius: 左上 右上 右下 左下
	//按左上,右上,右下,左下顺时针的方式指定四个角的半径
	border-radius: 圆角半径 //可以使用一个值，指定所有圆角半径
	border-radius: 左上右下圆角半径 右上左下圆角半径;
	//也可以使用两个值，指定对角半径

border-collapse*	border-collapse 属性设置表格的边框是否被合并为一个单一的边框，还是象在标准的 HTML 中那样分开显示。 separate 默认值。边框会被分开。不会忽略 border-spacing 和 empty-cells 属性 collapse 如果可能，边框会合并为一个单一的边框。会忽略 border-spacing 和 empty-cells 属性 inherit 规定应该从父元素继承 border-collapse 属性的值
border-color	参考 CSS
border-left	参考 CSS
border-left-color	参考 CSS
border-left-style	参考 CSS
border-left-width	参考 CSS
border-right	参考 CSS
border-right-color	参考 CSS
border-right-style	参考 CSS
border-right-width	参考 CSS
border-style	参考 CSS
border-top	参考 CSS
border-top-color	参考 CSS
border-top-style	参考 CSS
border-top-width	参考 CSS
border-width	参考 CSS
float	定义元素在哪个方向浮动。以往这个属性总应用于图像，使文本围绕在图像周围，不过在 CSS 中，任何元素都可以浮动。浮动元素会生成一个块级框，而不论它本身是何种元素。 如果浮动非替换元素，则要指定一个明确的宽度；否则，它们会尽可能地窄。 注释：假如在一行之上只有极少的空间可供浮动元素，那么这个元素会跳至下一行，这个过程会持续到某一行拥有足够的空间为止。 指定下面的值之一： none 默认。元素不浮动，并会显示在其在文本中出现的位置 left 元素向左浮动 right 元素向右浮动
margin	该属性可以有 1 到 4 个值。 这个简写属性设置一个元素所有外边距的宽度，或者设置各边上外边距的宽度。

	块级元素的垂直相邻外边距会合并，而行内元素实际上不占上下外边距。行内元素的的左右外边距不会合并。同样地，浮动元素的外边距也不会合并。允许指定负的外边距值，不过使用时要小心。 注释：允许使用负值。
margin-bottom	指定下面的值之一： auto 默认。底部外边距等于顶部外边距 height 浮点数字，后面跟个绝对长度单位(cm、mm、in、pt、pc、px) 或一个相对长度单位(em、ex) percentage 整数值，后面跟个百分号(%)。相对于父对象高度的百分比 %% 单位 占自由空间的百分比 calc(...) calc 函数
margin-left	--"--
margin-right	--"--
margin-top	--"--
padding	设置或获取元素与它的外边距间插入的空间尺寸(内边距)，如果有边框，则是对象与边框间的空间尺寸。 这个简写属性设置元素所有内边距的宽度，或者设置各边上内边距的宽度。行内非替换元素上设置的内边距不会影响行高计算；因此，如果一个元素既有内边距又有背景，从视觉上看可能会延伸到其他行，有可能还会与其他内容重叠。元素的背景会延伸穿过内边距。不允许指定负边距值。 指定下面的值之一： length 浮点数字，后面跟个绝对长度单位(cm、mm、in、pt、pc、px) 或一个相对长度单位(em、ex) percentage 整数值，后面跟个百分号(%)。相对于父对象宽度的百分比 %% 单位 占自由空间的百分比 calc(...) calc 函数
padding-bottom	--"--
padding-left	--"--
padding-right	--"--
padding-top	--"--

3.5 分类属性

list-style	指定 1 到 3 个下面的属性，可以任意顺序：type，position，image。
list-style-image	指定以下值之一： none 默认。没有指定图像 url(sURL) 图像的地址，其中 sURL 既可以是绝对也可以是相对 URL 地址
list-style-position	设置在何处放置列表项标记。 该属性用于声明列表标志相对于列表项内容的位置。外部（outside）标志会放在离列表项边框边界一定距离处，不过这距离在 CSS 中未定义。内部（inside）标志处理为好像它们是插入在列表项内容最前面的行内元素一样。 指定以下值之一： inside 列表项目标记放置在文本以内，且环绕文本根据标记对齐 outside 默认值。保持标记位于文本的左侧。列表项目标记放置在文本以外，且环绕文本不根据标记对齐
list-style-type	设置列表项标记的类型 指定下面的值之一： disc 默认。标记是实心圆 circle 标记是空心圆

	square 标记是实心方块 decimal 标记是数字，如：1, 2, 3, 4 等 lower-roman 小写罗马数字(i, ii, iii, iv, v, 等) upper-roman 大写罗马数字(I, II, III, IV, V, 等) lower-alpha 小写英文字母(a, b, c, d, e, 等) upper-alpha 大写英文字母(A, B, C, D, E, 等) tree-line 画树线 none 没有标记
list-marker-color	设置列表标记(实心圆(方块)、数字、树线)的颜色
list-marker-size	设置画实心圆(方块)、数字的字体尺寸。对于 list-style-type:tree-line 属性是设置树线的宽度
list-marker-style	对于 list-style-type:tree-line 属性，是设置树线的样式： none 默认。不画线，忽略边框宽度 dotted 点划线(Dotted) dashed 虚线(Dashed) solid 边框是实线(solid)

3.6 定位和维度属性

position	规定元素的定位类型。 这个属性定义建立元素布局所用的定位机制。任何元素都可以定位，不过绝对或固定元素会生成一个块级框，而不论该元素本身是什么类型。相对定位元素会相对于它在正常流中的默认位置偏移。 可能的值为： absolute 生成绝对定位的元素，相对于 static 定位以外的第一个父元素进行定位。元素的位置通过 "left", "top", "right" 以及 "bottom" 属性进行规定。 fixed 生成绝对定位的元素，相对于浏览器窗口进行定位。元素的位置通过 "left", "top", "right" 以及 "bottom" 属性进行规定。 relative 生成相对定位的元素，相对于其正常位置进行定位。因此，"left:20" 会向元素的 LEFT 位置添加 20 像素。 static 默认值。没有定位，元素出现在正常的流中（忽略 top, bottom, left, right 或者 z-index 声明）
left right top bottom	指定以下值之一： auto 默认 height 浮点数字，后面跟个绝对长度单位(cm、mm、in、pt、pc、px) 或一个相对长度单位(em、ex) percentage 整数值，后面跟个百分号(%)。相对于父对象高度的百分比，必须显示指定，不允许为负值 calc(...) calc 函数
height	指定以下值之一： auto 默认 height 浮点数字，后面跟个绝对长度单位(cm、mm、in、pt、pc、px) 或一个相对长度单位(em、ex) percentage 整数值，后面跟个百分号(%)。相对于父对象高度的百分比，必须显示指定，不允许为负值 %% 单位 占自由空间的百分比 width(NN%) 设置高度为该元素宽度的百分比值。该属性运行元素师保持比例缩放 calc(...) calc 函数
z-index	设置元素的堆叠顺序。拥有更高堆叠顺序的元素总是会处于堆叠顺序较低的元素的前面。

	注释：元素可拥有负的 z-index 属性值。 注释：Z-index 仅能在定位元素上奏效（例如 position:absolute;）！ 该属性设置一个定位元素沿 z 轴的位置，z 轴定义为垂直延伸到显示区的轴。如果为正数，则离用户更近，为负数则表示离用户更远。 可能的值： auto 默认。堆叠顺序与父元素相等 number 设置元素的堆叠顺序 inherit 规定应该从父元素继承 z-index 属性的值

overflow	规定当内容溢出元素框时发生的事情。它是 overflow-x 和 overflow-y 的简写形式。
	这个属性定义溢出元素内容区的内容会如何处理。如果值为 scroll，不论是否需要，用户代理都会提供一种滚动机制。因此，有可能即使元素框中可以放下所有内容也会出现滚动条。
	格式：overflow-type [scroll-manner]。
	第一个 overflow-type 的值为以下值：
	visible 默认值。内容不会被修剪，会呈现在元素框之外，不会出现滚动条
	scroll 内容会被修剪，但是浏览器会显示滚动条以便查看其余的内容
	hidden 内容会被修剪，并且其余内容是不可见的，不会出现滚动条
	auto 如果内容被修剪，则浏览器会显示滚动条以便查看其余的内容
	hidden-scroll 滚动条不显示，但是元素的内容是可以滚动的
	scroll-indicator 当鼠标经过该元素时显示滚动位置指示器
	none 等价于代码：
	{
	overflow:visible;
	min-width:min-content;
	min-height:min-content;
	}
	第二个参数(可选)是 scroll-manner()函数的定义，见下面的 scroll-manner 的定义。

overflow-x overflow-y	格式：overflow-type [scroll-manner]。 第一个 overflow-type 的值为以下值： visible 默认值。内容不会被修剪，会呈现在元素框之外，不会出现滚动条 scroll 内容会被修剪，但是浏览器会显示滚动条以便查看其余的内容 hidden 内容会被修剪，并且其余内容是不可见的，不会出现滚动条 auto 如果内容被修剪，则浏览器会显示滚动条以便查看其余的内容 hidden-scroll 滚动条不显示，但是元素的内容是可以滚动的 scroll-indicator 当鼠标经过该元素时显示滚动位置指示器 none 等价于代码： { overflow:visible; min-width:min-content; } 或者 { overflow:visible; min-height:min-content; } 第二个参数(可选)是 scroll-manner()函数的定义，见下面的 scroll-manner 的定义。
scroll-manner scroll-manner-x scroll-manner-y	这些属性可以接受 inherit 值，或者 scroll-manner()函数，scroll-manner()的参数可以为: inherit 默认值 animation true | false，启用/禁用 所有的****-animation 属性。 page-animation true | false，如果为 true,则 UP/DOWN 按键将引起平滑滚动**页**动画 step-animation true | false，如果为 true,则 UP/DOWN 按键将引起平滑滚动动画 home-animation true | false，如果为 true,则 HOME/END 按键将引起平滑滚动动画 wheel-animation true | false，如果为 true,则鼠标滚轮事件将引起平滑滚动动画 step length | percent | auto，一次滚动的步长(UP/DOWN 按键)的计算基于第

	一个可见元素的尺寸。滚动步长的 length 值将转换成像素值。percent 值将基于对应元素的尺寸计算。
	默认值是宽度或高度的 6.25% (1/16)。 auto 意味着"积分步长" - 滚动整行。
	page length | percent | auto，auto 意思是"积分步长" - 单次滚动步长(UP/DOWN 按键)的计算基于最后一个可见项的尺寸。滚动步长的 length 值将转换成像素值。percent 值将基于对应元素的尺寸计算。
	默认值是 100% - 元素的整个滚动区域的宽度或高度。
	滚动条手动滚动样式的定义示例：
	div#scrollable {
	overflow:auto;
	scroll-manner : scroll-manner(page-animation:true,
	step-animation:false,
	step:auto);
	}

width	指定下面的值之一： auto 默认。元素的默认宽度 width 浮点数字，后面跟个绝对长度单位(cm、mm、in、pt、pc、px) 或一个相对长度单位(em、ex) percentage 整数值，后面跟个百分号(%)。相对于父对象宽度的百分比，不需要负值 %% 单位 占容器的自由空间的百分比 calc(...) calc 函数
min-height	指定元素的最小高度。 length 浮点数字，后面跟个绝对长度单位(cm、mm、in、pt、pc、px) 或一个相对长度单位(em、ex) percentage 整数值，后面跟个百分号(%)。相对于容器块高度的百分比作为元素的最小高度。如果包含块的高度没有显示指定，则该元素没有最小高度，该属性被解释成 0% width(NN%) 将会设置 min-height 为元素本身宽度的百分比值。这种值允许元素保持宽高比 calc(...) calc 函数
min-width	指定元素的最小宽度。 length 浮点数字，后面跟个绝对长度单位(cm、mm、in、pt、pc、px) 或一个相对长度单位(em、ex) percentage 整数值，后面跟个百分号(%)。相对于父对象宽度的百分比，不允许负值 auto min-width 等于元素的最小内在宽度。 在 HTMLayout 中，所有元素默认情况下都会获取这个值来模仿 IE 行为。非标准 calc(...) calc 函数
max-height	指定元素的最大高度 length 浮点数字，后面跟个绝对长度单位(cm、mm、in、pt、pc、px) 或一个相对长度单位(em、ex) percentage 整数值，后面跟个百分号(%)。相对于父对象高度的百分比。不允许负值 width(NN%) 将会设置 max-height 为元素本身宽度的百分比值。这种值允许元素保持宽高比 calc(...) calc 函数
max-width	指定元素的最大宽度 length 浮点数字，后面跟个绝对长度单位(cm、mm、in、pt、pc、px) 或一个相对长度单位(em、ex) percentage 整数值，后面跟个百分号(%)。相对于父对象高度的百分比。不允许负值 auto max-width 等于元素的最大内在宽度。

	在 HTMLayout 中，<TABLE>默认有这个值。它允许在未设置 width 属性时来决定 table 的尺寸。 非标准。 示例：边框看起来像是设置了 max-width:auto: `这个表(table)没有设置 width 属性` calc(...) calc 函数
flow	在块容器(如 DIV)中子元素的流式布局方向(Flow direction)。这个属性定义了块容器使用的布局管理器(layout manager)(LM)。 vertical 默认值。块元素中的所有子元素按照从上到下的位置摆放，是 HTML 中 DIV 的标准行为 horizontal 块元素中的所有子元素按照从左到右的位置摆放成一行 horizontal-flow 多行布局。 块元素中的所有子元素按照从左到右的位置摆放，当超出块边界时转换到下一行。在这个布局中 height:100%%等于一行的高度 vertical-flow 多列布局。 块元素中的所有子元素按照从上到下的位置摆放，当超出块边界时转换到下一列。在这个布局中 width:100%%等于一列的宽度 "...模板..." stack 标签页容器。子元素作为卡片堆叠在一起。子元素的渲染顺序由子元素的 z-index 属性定义。每个子元素的弹力单位的计算都是独立的 更多参见 HTMLayout flow 布局样式
hit-margin	指定下面的 1 到 4 个以空格分隔的值。语法于 margin 属性相同。hit-margin 定义元素的"悬停区域"。正数/负数 增加/减少 悬停区域。hit-margin 的计算从元素边框(border)盒开始。
size	length [length] 这个属性是 width、height 属性的简写形式。如果只提供了一个值，则 width 和 height 都等于这个值。如果提供了两个值，则第一个值是 width，第二个值是 height。
mapping	inherit | none | left-to-right | top-to-right | inherit(<part-list>) | none(<part-list>) | left-to-right(<part-list>) | top-to-right(<part-list>) 其中，<part-list>是一个以逗号分隔的列表，它包含了一个或多个下面的描述: border, padding, margin, background, foreground, layout。 该属性设置方向相关属性的映射。如下面的声明: `div { mapping: left-to-right(border,margin); }` 会导致边框(颜色、宽度、样式)和外边距的左和右进行交换(即镜像)，所以示例中左边框会当做右边框，右边框会当做左边框来绘制。 注意：该属性默认是可继承的。所以定义 body { mapping: left-to-right;}后，该元素和它的继承元素的方向属性都会进行左右镜像转换。<part-list>中名称的含义: border 边框的所有相关属性: color, style, width padding 所有内边距属性: padding-left, padding-right 等 margin 所有外边距属性: margin-left, margin-right, hit-margin 等 background 有方向的背景属性: background-position, background-image ，背景渐变 foreground 有方向的前景属性: foreground-position, foreground-image layout 例如：flow:horizontal; mapping: left-to-right(layout); 将会导致子元素从右到左的方向放置

3.7 轮廓(Outline)

轮廓与边框相似，不过它们有以下不同:

1. 轮廓不会占用空间。
2. 轮廓矩形的所有边界拥有相同的属性值

outline	outline （轮廓）是绘制于元素周围的一条线，位于边框边缘的外围，可起到突出元素的作用。 注释：轮廓线不会占据空间，也不一定是矩形。 outline 简写属性在一个声明中设置所有的轮廓属性。 可以按顺序设置如下属性： outline-color outline-style outline-width 如果不设置其中的某个值，也不会出问题，比如 outline:solid #ff0000; 也是允许的。 可以指定以下值: [<'outline-color'> || <'outline-style'> || [<'outline-width'> [<'outline-offset'>]]
outline-width	轮廓宽度： medium 默认值 thin 比默认宽度细 thick 比默认宽度粗 width 浮点数字，后面跟个绝对长度单位(cm、mm、in、pt、pc、px) 或一个相对长度单位(em、ex) calc(...) calc 函数
outline-style	指定以下值之一： none 默认值。不绘制边框，忽略边框宽度 dotted 轮廓是点划线(dotted) dashed 轮廓是虚线(dashed) solid 轮廓是实线 glow 轮廓有光晕效果
outline-color	轮廓的颜色
outline-offset	长度单位，轮廓的偏移量。正数值-向外偏移，负数值-向内偏移。CSS3 属性

3.8 伪类和其他属性

cursor	auto | crosshair | default | pointer | move | e-resize | ne-resize | nw-resize | n-resize | se-resize | sw-resize | s-resize | w-resize | text | wait | help | progress 值或.cur/.ani 类型文件的 URL
display	display 属性规定元素应该生成的框的类型。 这个属性用于定义建立布局时元素生成的显示框类型。对于 HTML 等文档类型，如果使用 display 不谨慎会很危险，因为可能违反 HTML 中已经定义的显示层次结构。对于 XML，由于 XML 没有内置的这种层次结构，所有 display 是绝对必要的。 可能的值： none 此元素不会被显示 block 此元素将显示为块级元素，此元素前后会带有换行符 inline 默认。此元素会被显示为内联元素，元素前后没有换行符 list-item 此元素会作为列表显示
visibility	visibility 属性规定元素是否可见。 这个属性指定是否显示一个元素生成的元素框。这意味着元素仍占据其本来的空间，不过可以完全不可见。 可能的值有：

	visible 默认值。元素可见且占用空间 hidden 元素不可见，但占用空间 collapse 元素不可见，且不占用空间 注意: HTMLayout 对 collapse 值进行了扩展，它不仅支持标准 CSS 中的表格(table)，也支持所有的其他元素。visibility: collapse 等同于 display:none，不过 collapsed 元素会参与容器的内在宽度的计算，因此动态修改 visibility:visible 成 visibility:collapse 不会引起容器元素的宽度计算改变 推荐通过 visibility:visible/collapse 来设置动态的隐藏/显示效果。通过设置 display:none 来隐藏元素是很容易的，但是重新显示它会比较麻烦——有多种 display 值可供选择: block，inline-block，list-item 等
direction	指定以下值之一: ltr 默认，内容从左到右布局 rtl 默认，内容从右到左布局 inherit 内容布局从父元素继承
transition	none | blend | slide | image 定义修改元素状态时的过渡效果。例如: .mybutton:hover { color: red; transition:blend } 将会通过不断地融合(混合)原始状态和悬停状态来实现过渡效果 当前，transition 仅支持 block、inline-block 元素和 table 元素

3.9 分页媒体(打印)

page-break-before	指定下面的值之一: auto 默认值 always 在该元素之前总是插入分页符 percentage 整数值，后面跟着一个百分号(%)。 **有条件分页**。该值是页高度的百分比值，如果元素在页上的位置大于这个值，则引擎会在它之前插入分页符
page-break-after	指定下面的值之一: auto 默认值 always 在该元素之后总是插入分页符 percentage 整数值，后面跟着一个百分号(%)。 **有条件分页**。该值是页高度的百分比值，如果元素的底部在页上的位置大于这个值，则引擎会在它之后插入分页符

3.10 行为属性

behavior	以空格分隔的 nmtoken 列表 - 应用到该元素上的行为名称列表。 可以在列表的前面后后面包含'~'标识符: behavior: ~ new-behavior 插入新行为前，会更新该元素之前定义的行为列表 behavior: new-behavior ~ 插入新行为到该元素之前定义的行为列表的后面 Behavior 是应用程序或内建的 behavior 类型的名称。(内建)标准行为列表参见 HTMLayout 内置 behavior 大全

3.11 其他属性

content	"string" | attr(attrname) 允许在样式中重定义元素的文本内容。在 h-smile 中，content 可以应用到元素它本身。 如果 content 值是一个字符串，则该字符串直接替换元素的内容。如果 content 的值是 attr(attrname)，则元素的内容替换为该元素 attrname 属性的值。 示例：这个属性允许作为一个开关给一些元素通过 CSS 定义不同的标题: p.status[state="pending"] { content: "Pending";foreground-image:...; } p.status[state="done"] { content: "Done"; foreground-image:...; } p.status[state="warning"] { content:"Warning!";foreground-image:...;} content 属性只允许应用于有文本内容的属性。例如，它对于 P、SPAN 等是有效的，但它对 DIV 是无效的。
vertical-scrollbar	"style-set-name" - 定义的垂直滚动条的样式集。
horizontal-scrollbar	"style-set-name" - 定义的水平滚动条的样式集。参见: html_samples/css-plus/scrollbar-styling.htm 文件中是怎么定义的。

3.12 长度单位

引擎支持以下几种长度单位：

px	像素，相对于视图设备。在屏幕上 1px 对应 1 设备像素。在打印机上 1px 对应 1/96 英尺
em	与字体相关的 *font-size*
ex	与字体相关的 *x-height*
in	英尺 — 1 英尺等于 2.54 厘米
cm	厘米
mm	毫米
pt	点 — 点是由 CSS2.1 使用的，1pt 等于 1/72 英尺
pc	派卡(Pica)。绝对长度单位。相当于我国新四号铅字的尺寸。1pc 等于 12pt
%	百分数 — 通常等于对应父元素的度量值的百分比值，但也不是总是这样，参见 CSS 规范
%%	flex ("weight")
*	flex ("weight") — 1* 等于 100%%
dip	设备独立像素(device independent pixels) - 逻辑单位. 1dip = 1/96(英尺)，因此在 96DPI 的显示器上等于 1 像素。在高 dpi 屏幕上，它将会大于物理像素。示例: 在 120DPI 屏幕上，2dip 将等于 3px。在打印机上，'dip'和'px'单位是相等的

3.13 颜色

引擎支持以下几种颜色常量的定义：

color: #f00	#rgb 形式
color: #ff0000	#rrggbb 形式
color: rgb(255,0,0)	rgb(0..255,0..255,0..255) 形式
color: rgb(100%,0%,0%)	rgb(0..100%,0..100%,0..100%) 形式
color: hsl(300,0%,0%)	rgba(0..255,0..255,0..255,0..1) 形式
color: tint(@base-color, 50%, 0%)	tint(base-color, -1.0..1.0 [, -1.0..1.0]) 形式

1. tint(src-color, luminance-delta, saturation-delta)允许基于基础色定义颜色。参数：
   src-color 是一个颜色描述或者一些预定义的颜色常量。

2. luminance-delta 是范围-1.0 .. +1.0 的浮点值 - 对原始颜色亮度的相对增加值:
如果 luminance-delta < 0 则 final-luminance = src-luminance * abs(luminance-delta)
如果 luminance-delta > 0 则 final-luminance = (1.0 - src-luminance) * luminance-delta

3. saturation-delta, 可选参数, 是范围-1.0 .. +1.0 的浮点值 - 对原始颜色色彩饱和度的相对增加值:
如果 luminance-delta < 0 则 final-luminance = src-luminance * abs(luminance-delta)
如果 luminance-delta > 0 则 final-luminance = (1.0 - src-luminance) * luminance-delta

示例，请看下面的声明:

```
@const BASE_COLOR: #FF0000; /*红色*/
li { border:1px solid tint( @BASE_COLOR, -0.5); }
<li>元素的边框将会是暗红色 - #7F0000。
```

三、behavior 大全

1.clickable 响应单击节点

事件	onButtonClick onButtonPress
备注	clickable 用于实现支持 onButtonClick，onButtonPress 单击事件但是不支持焦点切换的按钮，例如工具条按钮。

代码	import win.ui;
	/*DSG{{*/
	winform = win.form(text="HTMLayout behavior:clickable";right=599;bottom=399)
	/*}}*/
	
	import web.layout;
	wbLayout = web.layout(winform);
	
	//下面的自定义 behavior 追加到节点上，用于处理事件
	namespace web.layout.behavior.eventhandler {
	
	onButtonClick = function (ltTarget,ltOwner,reason,behaviorParams) {
	ltOwner.innerText = "单击事件"
	}
	
	onButtonPress = function (ltTarget,ltOwner,reason,behaviorParams) {
	ltOwner.innerText = "当前节点得到焦点时,按下鼠标左键或空格键"
	}
	
	onMouseClick = function(ltTarget,ltOwner,x,y,ltMouseParams) {
	ltOwner.innerText = "behavior 中移除 clickable 以后单击触发 onMouseClick 事件而不是 onButtonClick"
	}
	}
	
	wbLayout.html =/***

	请点击这里 ***/ winform.show() win.loopMessage();

2.edit 文本框控件

HTML	<input type="text" /> <widget type="text"></widget>
属性	value=“text” - 文本框初始值 size=integer - 文本框宽度 maxlength=integer -可包含的最大字符数 filter=“filter-expr” - 限制输入的字符表达式，可指定单个字符或指定字符范围，例如 “.@0~9a~zA~Z”表示允许所有数字字母，'.' 以及 '@'. 可以使用 '^' 前缀排除字符. 例如“^.,-” 允许所有除 '.', ',' 和 '-' 以外的字符。 novalue=“text” - 文本框为空时显示的默认值，在 CSS 中可以使用:empty 指定显示默认值的样式。
事件	onEditValueChanged 控件值已变更 onEditValueChanging 正在变更控件值
快捷键	LEFT，CTRL+LEFT，SHIFT+LEFT，CTRL+SHIFT+LEFT RIGHT，CTRL+RIGHT，SHIFT+RIGHT，CTRL+SHIFT+RIGHT HOME，SHIFT+HOME END，SHIFT+END BACKSPACE，ALT+BACKSPACE，CTRL+BACKSPACE CTRL+A，CTRL+X，CTRL+V，CTRL+Z DELETE，SHIFT+DELETE，CTRL+DELETE INSERT，SHIFT+INSERT，CTRL+INSERT CTRL+(LEFT)SHIFT 以及 CTRL+(RIGHT)SHIFT
自定义函数	ltEle.xcall("selectionStart"): int - 返回选区开始位置 ltEle.xcall("selectionEnd"): int - 返回选区结束位置 ltEle.xcall("setSelection", start:int, end:int):void - 设置选区 ltEle.xcall("selectionText"): string - 返回选中文本 ltEle.xcall("insertText", text: string):void - 在光标处插入文本 ltEle.xcall("appendText", text: string):void - 追加文本 ltEle.xcall("undo", false): bool - 是否可以撤消操作 ltEle.xcall("undo", true) - 撤消操作 ltEle.xcall("cut", false): bool - 是否可以剪切 ltEle.xcall("cut", true) - 剪切操作 ltEle.xcall("copy", false): bool - 是否可复制 ltEle.xcall("copy", true) - 复制操作 ltEle.xcall("paste", false): bool - 是否可粘贴 ltEle.xcall("paste", true) - 粘贴操作; ltEle.xcall("selectAll"): bool - 当前是否已全选文本 ltEle.xcall("selectAll", true) - 全选文本

3.plaintext 文本框控件

HTML	<textarea>...</textarea> – display:inline-block editor; <plaintext>...</plaintext> – display:block editor;
属性	value=“text” - 文本框初始值 maxlength=integer -可包含的最大字符数 filter=“filter-expr” - 限制输入的字符表达式，可指定单个字符或指定字符范围，例如 “.@0~9a~zA~Z”表示允许所有数字字母，'.' 以及 '@'. 可以使用 '^' 前缀排除字符. 例如“^.,-” 允许所有除 '.', ',' 和 '-' 以外的字符。 novalue=“text” - 文本框为空时显示的默认值，在 CSS 中可以使用:empty 指定显示默认值的样式。readonly 指定该属性则显示为只读模式。
事件	onEditValueChanged 控件值已变更 onEditValueChanging 正在变更控件值
快捷键	LEFT，CTRL+LEFT，SHIFT+LEFT，CTRL+SHIFT+LEFT RIGHT，CTRL+RIGHT，SHIFT+RIGHT，CTRL+SHIFT+RIGHT HOME，SHIFT+HOME END，SHIFT+END BACKSPACE，ALT+BACKSPACE，CTRL+BACKSPACE CTRL+A，CTRL+X，CTRL+V，CTRL+Z DELETE，SHIFT+DELETE，CTRL+DELETE INSERT，SHIFT+INSERT，CTRL+INSERT CTRL+(LEFT)SHIFT 以及 CTRL+(RIGHT)SHIFT
自定义函数	ltEle.xcall("selectionStart"): int - 返回选区开始位置 ltEle.xcall("selectionEnd"): int - 返回选区结束位置 ltEle.xcall("setSelection", start:int, end:int):void - 设置选区 ltEle.xcall("selectionText"): string - 返回选中文本 ltEle.xcall("insertText", text: string):void - 在光标处插入文本 ltEle.xcall("appendText", text: string):void - 追加文本 ltEle.xcall("canUndo"): bool - 是否可以撤消操作 ltEle.xcall("doUndo") - 撤消操作 ltEle.xcall("canRedo"): bool - 是否可以重做操作 ltEle.xcall("doRedo") - 重做操作 ltEle.xcall("canCut"): bool - 是否可以剪切 ltEle.xcall("doCut") - 剪切操作 ltEle.xcall("canCopy"): bool - 是否可复制 ltEle.xcall("doCopy") - 复制操作 ltEle.xcall("canPaste"): bool - 是否可粘贴 ltEle.xcall("doPaste") - 粘贴操作; ltEle.xcall("canSelectAll"): bool - 当前是否已全选文本 ltEle.xcall("canSelectAll") - 全选文本

4.decimal，number 数值输入框控件

HTML	<input type="decimal" /> <widget type="decimal"></widget> <input type="number" /> <widget type="number" ></widget>
属性	value=float - 初始值

	size=integer - 控件宽度
	min - 最小值
	max - 最大值
	step - 滚动选框步进值，指定了此属性会显示滚动选框
	format - 数值显示格式，支持下面的字段，使用 ';'分隔多个字段:
	grouping - number, '千位' 分隔符间隔数
	fdigits - number, 小数位置
	leading-zero - true/false, 是否显示前导 0
	decimal-sep - 小数分隔符
	grouping-sep - 千位分隔符
	negative-sign - true/false, true - 是否显示负号
	
	示例:
	format="grouping:3;
	fdigits:4;
	leading-zero:false;
	decimal-sep:',';
	thousand-sep:'_';
	negative-sign:false"
	
	也可以写在 CSS 自定义属性中(属性名前面加一个短横线)
	input.decimal
	{
	behavior:decimal;
	-format="fdigits:2; leading-zero:true;"
	}

事件	onEditValueChanged 控件值已变更 onEditValueChanging 正在变更控件值
自定义函数	ltEle.xcall("min"): int - 返回最小值 ltEle.xcall("min", v:int) - 设置最小值 ltEle.xcall("max"): int - 返回最大值 ltEle.xcall("max", v:int) - 设置最大值 ltEle.xcall("step"): int - 返回步进值 ltEle.xcall("step", v:int|undefined) - 设置|或清除步进值，设置了该值以后后会显示滚动选框

5.password 密码输入框控件

HTML	 <widget type="password"></widget>
属性	password-char="占位符" - 密码控件实际上是一个 edit 文本框控件，然后增加支持这个属性

6.currency 货币数值输入框控件

HTML	<input type="currency" /> <widget type="currency"></widget>
属性	value=float - 初始值

	size=integer - 控件宽度
	format - 数值显示格式，支持下面的字段，使用 ';'分隔多个字段:
	grouping - number, '千位' 分隔符间隔数
	fdigits - number, 小数位置
	leading-zero - true/false, 是否显示前导 0
	decimal-sep - 小数分隔符
	grouping-sep - 千位分隔符
	negative-sign - true/false, true - 是否显示负号
	currency - 货币符号
	currency-after - 货币符号是否显示在数值后面
	
	currency 控件会自动更新下面两个属性
	invalid - 当前输入了无效的值
	negative - 负数
	
	format 也可以写在 CSS 自定义属性中(属性名前面加一个短横线)
	input.decimal
	{
	behavior:decimal;
	-format="fdigits:2; leading-zero:true;"
	}

事件	onEditValueChanged 控件值已变更 onEditValueChanging 正在变更控件值

7.button 按钮控件

HTML	<button></button> <input type="button" /> <widget type="button"></widget> <input type="reset" /> <input type="submit" /> <button> 可以使用 HTML 指定按钮标题</botton>
属性	value="caption" - 按钮标题
事件	onButtonClick - 单击事件 onButtonPress - 当前节点得到焦点时,按下鼠标左键或空格键

8.check 复选框控件

HTML	<input type="checkbox"> <widget type="checkbox"> <button type="checkbox">caption</button> <input type="checkbox" mixed> - 三态选框,inline 节点。 <widget type="checkbox" mixed> - 三态选框,block 节点。 <button type="checkbox" mixed>Windows like inline checkbox button</button> 三态选框
属性	checked 用于指定初始值，运行时的值应使用 ltEle.state.checked 获取。 checked=true - 选中

	checked=false - 取消选中 checked=undefined - 三态按钮的未决状态
事件	onButtonStateChanged - 控件值变更事件 onButtonPress - 当前节点得到焦点时,按下鼠标左键或空格键

9.radio 单选按钮

HTML	<input type="radio"> <widget type="radio"> <button type="radio">caption</button>
属性	checked 用于指定初始值，运行时的值应使用 ltEle.state.checked 获取。 checked=true - 选中 checked=false - 取消选中
事件	onButtonStateChanged - 控件值变更事件 onButtonPress - 当前节点得到焦点时,按下鼠标左键或空格键

10.switch 单选按钮

备注	behavior 与 behavior:radio 用法类似 主要区别是在鼠标左键按下时（而不是放开时）切换选中状态

11.progress 进度条控件

HTML	<progress /> - inline 节点 <input type="progress" /> - inline 节点 <widget type="progress"></widget> - block 节点
备注	progress 控件根据进度百分比调整前景图显示宽度以显示当前进度值。 关于指定前景图请参考：CSS 属性 background-position ,foreground-position 用法
属性	value=float - 进度值 maxvalue=float - 最大值 bar="clip" 如果指定这个 HTML 属性，或者在 CSS 中指定自定义属性 -bar:"clip"；进度条会剪切显示前景图 （始终不改变原图大小），这时候右侧显示为直边。如果不指定 clip 模式则前景图被拉伸显示（动态改变原图大小以适应显示宽度），请按九宫格指定前景切图参数（以保持右侧切片固定显示并拉伸中间部分）
事件	onButtonStateChanged - 控件值变更事件 onButtonPress - 当前节点得到焦点时,按下鼠标左键或空格键

12.time 时间控件

HTML	<input type="time" value="14:15:00" /> <widget type="time" value="14:15:00" ></widget>
属性	value= “HH:MM:SS” 或者 value= “now” - HH 使用 24 小时制，now 为当前时间 no-seconds - 是否需要秒数
事件	onEditValueChanged 控件值已变更 onEditValueChanging 正在变更控件值

13.slider 滑块控件

HTML	<input type="hslider"> - 水平滑块, inline block; <widget type="hslider"> - 水平滑块, block; <input type="vslider"> - 竖向滑块, inline block; <widget type="vslider"> - 竖向滑块, block;
属性	value - integer, 初始值. min - integer, 最小数值 max - integer, 最大数值 step - integer, 步进值,默认为 1 buddy - string, 同步显示控件值的节点 ID
事件	onButtonStateChanged - 控件值变更事件 onButtonClick - 单击控件

14.tree 树形视图控件

HTML	<select type="tree">…</select> <select type="tree" checkmarks>…</select> <widget type="tree">…</widget> <widget type="tree" checkmarks>…</widget>
备注	这个 behavior 与 behavior:select 很像，例如同样可以包含以下子节点： <option>…</option> - 表示一个列表项. <option> 还可以嵌套包含更多的列表项。 <optgroup>…</optgroup> - 列表项分组 控件的值返回选中的 option 的 value 属性或者 text 文本值。
属性	size="N" N 是大于 1 的数，指定显示行数，显示 1 行时为下拉列表，显示多行列表框 checkmarks - 是否显示复选框，其外观在 master CSS 中定义，可在 CSS 中自行义外观
事件	onSelectSelectionChanged - 控件选项变更 onSelectStateChanged - 展开或折叠子节点，ltTarget 为状态变更节点 节点 ltTarget.state.collapsed ，ltTarget.state.expanded 状态变更
快捷键	LEFT - 折叠子节点 RIGHT - 展开子节点 UP / PAGE-UP - 上一节点 DOWN / PAGE-DOWN- 下一节点 SHIFT + UP/PAGE-UP - 上一节点并且选中 SHIFT + UP/PAGE-DOWN - 下一节点并且选中 HOME - 第一个节点 END - 最后一个节点

15.select 列表控件

HTML	<select size=N>…</select> ，N 是大于 1 的数，指定显示行数，显示 1 行时为下拉列表，显示多行列表框 <select type=“select”>…</select> 注意这里即使 size=1 也会显示为 select，而不是下拉框（dropdown-select） <widget type=“select”></widget>
备注	select 控件内部可以包含任意的 HTML 子节点，注意以下子节点的用法： <option>…</option> - 表示一个列表项. <option> 还可以嵌套包含更多的列表项。 <optgroup>…</optgroup> - 列表项分组

	注意 behavior:select 仅仅是一个多功能的列表控件， behavior:dropdown-select 才是下拉框（“dropdown” 即“下拉”的意思） HTML 代码 <select size=1>…</select> 时加载 behavior:dropdown-select ，当 size 属性大于 1，或者在属性中指定了 type=“select”才会加载 behavior:select。 behavior:select 可以创建列表框(listbox)，也支持多列可以变成列表视图（listview），并且支持嵌套多级子节点变成树形视图(treeview，行为类似 behavior:tree ） select 控件可以实现下拉框(combobox),多行列表框(listbox)、列表视图(listview)以及树视图(treeview)，关于这个控件的用法与示例代码请参考文章：HTMLayout 使用 select 控件实现列表视图(listview)
属性	size=“N” - N 是大于 1 的数，指定显示行数，显示 1 行时为下拉列表，显示多行列表框 multiple 是否允许选中多个列表项 multiple=“checks” - 显示复选框
事件	onSelectSelectionChanged - 控件选项变更 onSelectStateChanged - 节点状态变更，例如树形节点展开或折叠子节点
快捷键	LEFT - 折叠子节点 RIGHT - 展开子节点 UP / PAGE-UP - 上一节点 DOWN / PAGE-DOWN- 下一节点 SHIFT + UP/PAGE-UP - 上一节点并且选中 SHIFT + UP/PAGE-DOWN - 下一节点并且选中 HOME - 第一个节点 END - 最后一个节点

16.dropdown-select 下拉框控件

HTML	<select>…</select> <select size=1>…</select> <select type="select-dropdown">…</select> <widget type="select-dropdown">…</widget>
备注	这个控件实际上是 select 控件显示为下拉框时应用的 behavior， 此 behavior 将节点创建为以下结构(CSS 写样式要按下面的结构修改子节点外观)： <select> <caption>标题</caption> <button>按钮</button> <popup> <option>列表项</option> <option>列表项</option> <option>列表项</option> </popup> </select> 子节点可以包含任意的 HTML,其中<option>节点或节点 HTML 属性中指定了 role="option"的节点视为子节点，例如： <select> <table> … <td role=option>First</td> <td role=option>Second</td>

	... </table> </select>
属性	editable - 如果为真,标题子节点 caption 将应用 behavior:edit,并且控件的 value 属性值也指向 edit 控件的当前值。 novalue="text" - 没有选中项时显示的默认值
事件	onSelectSelectionChanged - 控件选项变更 onSelectStateChanged - 节点状态变更，例如树形节点展开或折叠子节点
快捷键	UP / PAGE-UP - 上一节点 DOWN / PAGE-DOWN- 下一节点 HOME - 第一个节点 END - 最后一个节点 CTRL + DOWN - 显示下拉列表

17.form 表单

HTML	<form>
属性	action= “url” - 提交地址 target= “frame-name” - 目标框架 method= “get” | “post” - 提交方法 enctype= “application/x-www-form-urlencoded” | “multipart/form-data” - 编码类型
方法	ltEle.xcall("submit") - 提交表单 ltEle.xcall("reset") - 重置表单

事件	onFormSubmit = function (ltTarget,ltOwner,reason,behaviorParams) {
	var formData = behaviorParams.data.getValue();
	}
	onFormReset - 重置表单

18.scrollbar 滚动条

HTML	<input type="vscrollbar" /> <widget type="vscrollbar" /> - 竖向滚动条 <input type="hscrollbar" /> <widget type="hscrollbar" /> - 横向滚动条
事件	onScrollPos onScrollHome onScrollEnd onScrollStepPlus onScrollStepMinus onScrollPagePlus onScrollPageMinus onSliderRelease

19.hyperlink 超链接

HTML	
事件	onHyperlinkClick

	本事件较特殊，必须写在 sinking 过程里，具体请参考文章：HTMLayout 事件捕获与冒泡过程

代码	import process;
	wbLayout.sinking = {
	onHyperlinkClick = function (ltTarget,ltOwner,reason,behaviorParams) {
	process.execute(ltTarget.href);
	return true;//阻止事件流继续传递
	}
	}

20.menu 菜单

HTML	<menu class="popup"> <menu class="context">
备注	子节点中的 - 节点或 HTML 属性中指定了 role="menu-item"作为菜单项处理 菜单项也可以是如下的子菜单: <menu .popup> - Open - Close </menu>
事件	onMenuItemActive -鼠标位于某菜单项上,ltTarget 为激活的菜单项 onMenuItemClick -点击菜单项，ltTarget 为点击的菜单项

21.menu-bar 菜单栏

HTML	<ul #menu-bar> - Menu item with sub-menu <menu #sub-menu> - Open - Close </menu> - Simple menu item (no submenu)
备注	需要在 CSS 中指定 behavior:menu-bar,子节点中的 节点或 HTML 属性中指定了 role="menu-item"作为菜单项处理
事件	onMenuItemActive -鼠标位于某菜单项上,ltTarget 为激活的菜单项 onMenuItemClick - 点击菜单项，ltTarget 为点击的菜单项

22.popup-selector 弹出菜单下拉框

HTML	<button type="selector"> <caption>Select some value</caption> <menu> <li value="1">First <li value="2">Second ...

	</menu> </button>
备注	behavior: popup-selector 也可以实现下拉列表框, 区别是 popup-selector 是弹出一个菜单，并且多个 popup-selector 可以共用同一个弹出菜单。
属性	menu=“css-selector” - 使用 CSS 选择器指定要弹出的菜单，如果不指定此属性则使用子节点中的第一个<menu>或<popup>子节点 value=“string” - 初始值,菜单的 li 节点可以使用 value 属性指定值，点击菜单项时复制值到控件的<caption>标题节点,
事件	onButtonStateChanged 控件值变更

23.popup-menu 弹出菜单

备注	这个 behavior 类似 behavior: popup-selector 用于弹出一个菜单。 不同的是 popup-selector 主要是用来选中一个值（行为类似下拉列表框）， 而 popup-menu 仅仅是单纯的弹出菜单，接收的事件也是菜单事件，没有控件选中值的概念。 一个类似的 behavior 还有标准库中的 behavior:dropdown 也是用来弹出节点，不过 dropdown 弹出的不是菜单页只是普通的 popup 节点，弹出节点的方法有很多，详细的请参考文章：HTMLayout 使用 popup 弹出节点
属性	menu=“css-selector” - 使用 CSS 选择器指定要弹出的菜单，如果不指定此属性则使用子节点中的第一个<menu>或<popup>子节点 align-popup= left | top | right | bottom 用于指定菜单的弹出方向，默认是 bottom，也可以在 CSS 自定义属性中指定此值。
事件	onMenuItemActive -鼠标位于某菜单项上,ltTarget 为激活的菜单项 onMenuItemClick - 点击菜单项，ltTarget 为点击的菜单项

24.path-select 文件路径选框

HTML	<input type="file-path" /> - inline single line <widget type="file-path"></widget> <input type="folder-path" /> - inline single line <widget type="folder-path"></widget>
属性	filter="filename-filter" - 定义文件类型，多个类型分组用';'分隔，每个类型分组使用 ':'分隔标题以及后缀名列表。 <input name="uploadedfile" type="file" filter="HTML files:*.htm,*.html;Text files:*.txt;All files:*.*" novalue="(select file)"/> novalue="text" - 指定控件值为空时显示的默认文本
事件	onButtonStateChanged 控件值变更

25.file 文件上传控件

HTML	<input type="file" /> - inline single line <widget type="file"></widget>
备注	属性、事件、用法与上面的 behavior:path-select 类似， 但 behavior:file 实现的是标准 file 控件，放在<form enctype="multipart/form-data"> 里面可用于远程提交文件。 behavior:file 控件的值是只读的只能由用户在界面上指定（避免未经许可上传用户文件），而 behavior:path-select 的值可以编程修改。

	behavior:path-select 则用于本地程序他的值只是一个文件路径 - 并且可以编程修改。

26.frame 框架

HTML	<iframe src="something.htm"/>
属性	src="url.html" - 框架内加载网页地址 content-style="url.css" - 添加样式到框架页面 可以在 CSS 中使用 :busy 状态指定正在加载框架时的样式
方法	ltEle.xcall("load", url:string) - 在框架中加载指定地址网页 ltEle.xcall("clear") - 清空框架
事件	onFrameDocumentComplete 框架内页面加载完成

代码	import win.ui;
	/*DSG{{*/
	winform = ..win.form(text="HTMLayout 框架演示";bottom=399;parent=...;right=599;border="resizable")
	winform.add(
	layoutWindow={ dr=1;dl=1;notify=1;right=580;left=10;dt=1;top=12;z=1;db=1;bottom=382;multiline=1;cls="edit" }
	)
	/*}}*/
	
	import web.layout;
	wbLayout = web.layout(winform.layoutWindow);
	
	string.save("/frame1.html","frame1")
	string.save("/frame2.html","frame2")
	
	wbLayout.html =/***
	<button>点这里加载 frame2.html</button>
	
	  <iframe src="frame1.html" width="100%" height="100%">
	
	***/
	
	wbLayout.onButtonClick = function (ltTarget,ltEle,reason,behaviorParams) {
	  var ltFrame = wbLayout.queryEle("#main iframe");
	  ltFrame.xcall("load","frame2.html"); //加载网页到框架
	}
	
	winform.show()
	win.loopMessage();

27.column-resizer 可拖动调整列

备注	这个 behavior 专用于 table 节点使表格的标题列可以拖动调整大小，相应的 table 标记内必须使用 fixedrows=1 属性指定首行不随滚动条移动（作为表格的标题栏）。
代码	import win.ui;

	/*DSG{{*/ var winform = win.form(bottom=399;parent=...;right=599;text="behavior:column-resizer") /*}}*/ import web.layout; var wbLayout = web.layout(winform) wbLayout.html = /** <style> table { background:#CCC;margin:10px;} th{ background:#C12299;color:#FFF;min-width:180px; } td{ background:#EEE;} </style> <table border=0 cellspacing=1 width=100% fixedrows=1 style="behavior:column-resizer;"> <tr><th>可拖动调整大小</th><th>测试</th></tr> <tr> <td>测试</td> <td>测试</td> </tr> <tr> <td>测试</td> <td>测试</td> </tr> <tr> <td>测试</td> <td>测试</td> </tr> <tr> <td>测试</td> <td>测试</td> </tr> </table> **/ winform.show() win.loopMessage();

28.tabs 选项卡控件

HTML	

CSS	.tabs
	{
	behavior:tabs; /*指定这是一个选项卡控件*/
	}

备注	首先在 aardio 中事先导入这个 behavior，如下：
	import web.layout;
	import web.layout.behavior.tabs; //导入 behavior
	在选项卡控件里添加选项卡按钮，注意按钮一定要放在 CSS 类名为"strip"的 div 容器里
	
	<div panel="panel1" selected>选项一
	 选项二
	 选项三
	
	这是第一个选项页,其中 name 属性指定选项卡名字

```
        这是第二个选项页,其中 name 属性指定选项卡名字
        <button>按钮</button>
    </div>
    <div class="panel" name="panel3">
        这是第三个选项页,其中 name 属性指定选项卡名字
        <a href="tabs://http://www.baidu.com" target="_blank" title="百度">百度</a>
    </div>
  </div>
```

每个选项卡按钮里使用 panel 属性指定激活哪一个选项页面。

定义选项内容页面很简单，tabs 的直接子节点(不能是子节点下面的子节点)并指定 name 属性，并且 class="panel" 就行了，例如：

```
  <div class="panel"  name="panel3">
     这是第三个选项页,其中 name 属性指定选项卡名字
     <a href="tabs://http://wap.baidu.com" target="_blank" title="百度">百度一下</a>
  </div>
```

0tabs 选项卡中的页面有两种：

一种是固定的选项卡，一种是可以动态创建动态关闭的选项卡。

动态创建的选项卡有 colseable 属性，并且在选项卡上默认包含关闭按钮（需要用 CSS 自行指定关闭按钮的外观,参考示范代码）

在内容页中的超链接如果加上"tabs://"前缀，并指定 title,target 属性 - 那么在点击该节点时将会自动创建一个新选项卡打开，title 属性指定新建选项卡的标题，而 targer 指定目标选项卡的名字，如果 target 为"_blank"则表示创建新的选项卡。

也可以调用 tabs 提供的自定义函数创建或关闭页面，tabs 提供以下函数：

```
  ltTabs.xcall("select","选项卡名字")                                  选择一个页面
  ltTabs.xcall("close","选项卡名字")                                   关闭一个页面
  ltTabs.xcall("open  ","选项卡标题","框架页网址","选项卡名字")          打开一个页面
  ltTabs.xcall("create","选项卡标题","选项卡名字","子页面 HTML")        使用 HTML 代码新建一个页面
```

上面的 ltTabs 指的是 tabs 节点对象，新建的选项卡默认会添加 closeable 属性 - 可以使用代码或手动关闭。

固定的选项卡也可以用 open,close 函数显示或隐藏（但是不会删除页面），如果需要更详细的了解这些函数的用法可以查看标准库 web.layout.behavior.tabs 中这几个函数的定义。

最后就是添加修改 CSS 定义外观了，behavior 指定了行为（例如点哪个按钮激活对应的页面，响应事件等等），外观可以任意的定义，非常灵活，可以做出各种形形色色的选项卡。

29. popup 弹出节点

HTMLayout 扩展的 HTML 标记 - block 样式显示的块标记(类似)，用于定义可在页面上弹出显示的节点，默认的 popup 是隐藏不显示的。

一个常用的功能是使用<popup> 显示复杂些的工具提示(tooltips)

HTML 元素的 title 属性也会自动创建 popup 节点用于显示默认的提示信息：

```
<div title="这是提示" >鼠标放在这里</div>
```

修改这种内置 popup 节点的样式可使用如下 CSS:

```
_service > popup[role=tooltip]{ }
```

所有浏览器引擎自动创建的节点可以在 CSS 选择器指定父节点为 _service ，而 popup[role=tooltip] 匹配用于工具提示的 popup（这些提示节点的 HTML 属性中指定了 role="tooltip"）.

下面看一个完整的示例：

效果图


```
import win.ui;
/*DSG{{*/
var winform = win.form(text="HTMLayout - 气泡提示框 ";right=599;bottom=399)
/*}}*/

import web.layout;
var wbLayout = web.layout( winform )

wbLayout.html = /**
      <span title="这是一个漂亮的气泡提示框" style="margin:20px;">鼠标停在这里试试</span>
**/

wbLayout.css = /**
 _service > popup[role=tooltip]
  {
     border:0;
     width:max-intrinsic;
     font: 11pt "Comic Sans MS", Verdana;
     font-weight:normal;
     background-color:transparent;
     background-repeat: expand;
     padding:71px 25px 25px 16px;
     background-position:80px 33px 33px 74px; /*top right bottom left offsets*/
     background-image:url(
iVBORw0KGgoAAAANSUhEUgAAAKsAAACoCAYAAACffB63AAAACXBIWXMAAA7DAAAO
.....省略.....mMco0NIAAAAASUVORK5CYII=);
}
**/

winform.show()
win.loopMessage();
```

在指定 popup 样式中需要注意（menu 节点有类似问题）：popup 如果指定背景色会自动加上阴影（不能使用 outline 调整，如果指定圆角就会在右上角显示阴影内角），所以需要在 CSS 中指定 background-color:transparent; 清除背景色，然后自定义一

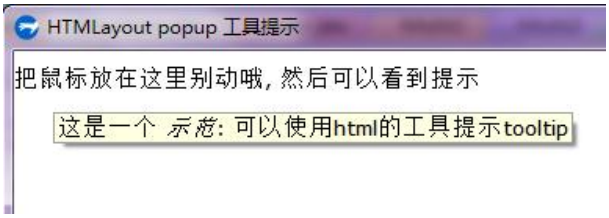
个背景图像即可（背景图像不会自动添加阴影，可以使用 png 图像自己画阴影上去），或者在 popup 里面用一个子节点指定背景色（避免 HTMLayout 添加阴影）。

在需要显示 popup 的节点上指定 titleid，那么当鼠标悬停在该节点上时，会使用指定 ID 的 popup 节点代替默认的 title 提示。

```
<p titleid="my-html-title">把鼠标放在这里别动哦，然后可以看到提示</p>
<popup id="my-html-title">
这是一个 <em>示范</em>:
可以使用 html 的工具提示 tooltip
</popup>
```

下面再看一个完整的例子下面是一个简单的 aardio 范例：

效果图




```
import win.ui;
/*DSG{{*/
var winform = ..win.form( bottom=399;parent=...;text="HTMLayout popup 工具提示";right=599 )
winform.add( )
/*}}*/

import web.layout;
wbLayout = web.layout(winform)

wbLayout.html = /**
<p titleid="my-html-title">把鼠标放在这里别动哦，然后可以看到提示</p>

<popup id="my-html-title">
这是一个 <em>示范</em>:
可以使用 html 的工具提示 tooltip
</popup>
**/

winform.show()
win.loopMessage();
```

如果我们需要在鼠标按下时弹出 popup 节点（ 而不是鼠标悬停在上面时显示 ）

那么需要使用一个 behavior:dorpdown，首先使用 import web.layout.behavior.dropdown; 自标准库导入这个 behavior，然后在需要弹出节点的对象上指定 behavior:dorpdown，并且在该节点内部增加一个 <popup> </popup> 子节点。

下面是一个简单的示例：

效果图

```
import win.ui;
/*DSG{{*/
var winform = ..win.form( bottom=399;parent=...;text="HTMLayout behavir:dropdown 演示";right=599 )
winform.add( )
/*}}*/

import web.layout;
wbLayout = web.layout(winform)

import web.layout.behavior.dropdown;

wbLayout.html = /**
<button #pop-button>点击这里弹出 popup 子节点
    <popup>
        这是一个 popup 子节点，默认他是隐藏的:
        <button>按钮</button>
    </popup>
</button>
**/

wbLayout.css = /**
#pop-button{
behavior:dropdown;
}
**/

winform.show()
win.loopMessage();
```

在上面的示例中 - 如果你在弹出的 popup 外面点击他就会自动隐藏，但是你在 popup 节点自身上点击鼠标他并不会隐藏。

如果希望在 popup 自身上点击也隐藏 popup，那就要在 popup 上添加 behavoir:popup，同样的 popup 也是标准库实现的 behavior(HTMLayout 并没有名字是 popup 的内部 behavior - 虽然他确实有一个叫 popup 的 HTML 标记)，所以首先要调用 import web.layout.behavior.popup;

```
import win.ui;
/*DSG{{*/
var winform = ..win.form( bottom=399;parent=...;text="HTMLayout behavior:popup 演示";right=599 )
winform.add( )
/*}}*/

import web.layout;
wbLayout = web.layout(winform)

import web.layout.behavior.dropdown;
import web.layout.behavior.popup;
```

```
wbLayout.html = /**
<button #pop-button>点击这里弹出节点
    <popup>
        这是一个 <em>弹出节点</em>:
       鼠标点击 关闭自己
    </popup>
</button>
**/

wbLayout.css = /**
#pop-button{
behavior:dropdown;
}
#pop-button popup{
behavior:popup;
}
**/

winform.show()
win.loopMessage()
```

如果希望点击 popup 节点不关闭，但是点击 popup 上的按钮关闭，可以使用 CSSS!来实现，如下：

效果图


```
import win.ui;
/*DSG{{*/
var winform = ..win.form( bottom=399;parent=...;text="HTMLayout behavior:popup 演示";right=599 )
winform.add( )
/*}}*/

import web.layout;
wbLayout = web.layout(winform)

import web.layout.behavior.dropdown;

wbLayout.html = /**
<button #pop-button>点击这里弹出节点
    <popup>
        这是一个 <em>弹出节点</em>:
        <button .close-popup>关闭自己</button>
    </popup>
</button>
**/
```

```
wbLayout.css = /**
#pop-button{
    behavior:dropdown;
}
popup button.close-popup{
    click!:
        self.$1p(popup):popup = false;
}
**/
winform.show()
win.loopMessage()
```

CSSS!	click!:
	self.$1p(popup):popup = false;
	
	self.$1p(popup) //函数向上查找第一个 popup 父节点，
	self.$1p(popup):popup = false //是修改节点的 popup 状态为 flase - 也就是隐藏弹出节点。

上面使用 popup 弹出节点有一个小问题，如果 popup 上有按钮，并且 popup 的位置超出了主窗体 - 那么在窗体范围外的部分点击按钮时不会重绘（看不到点击效果）

要解决这个问题，就可使用 behavior:popup-menu，popup-menu 是一个内建的 behavior，使用时无需导入，这个 behavior 通常用于弹出一个菜单（menu 节点 ），但也可以用于弹出一个普通的 popup 节点 - 这时候他的作用类似于标准库里的 dropdown，不过使用这个 behavior 弹出的节点没有前面所说的问题（ popup 上的按钮不能超出 窗体范围）

下面是使用 popup-menu 弹出节点的示例代码：
效果图


```
import win.ui;
/*DSG{{*/
var winform = ..win.form( bottom=164;parent=...;text="HTMLayout behavior:popup-menu 演示";right=184 )
winform.add(  )
/*}}*/

import web.layout;
wbLayout = web.layout(winform)

import web.layout.behavior.dropdown;

wbLayout.html = /**
<button #pop-button>点击这里弹出节点
    <popup>
        这是一个 <em>弹出节点</em>:
        <button>.....测试</button>
    </popup>
```

```
</button>
**/

wbLayout.css = /**
#pop-button{
    behavior:popup-menu;
}
#pop-button popup{
    behavior:menu;
}
**/

winform.show()
win.loopMessage()
```

需要注意按钮内的 popup 子节点需要在 CSS 中指定 behavior:menu; 否则 - 在按钮外面点击一下窗口范围外的按钮又无法重绘了。

另外 HTMLayout 的 behavior: popup-selector 也有类似 popup-menu 的作用。

popup-selector 实现的是下拉列表框，与 popup-menu 不同的他可以点击子节点切换选项值。
这个 popup-selector 如果使用 <button type="selector" menu="popup#some-popup" /> 这种方式使用时会添加一个向下的箭头，而且按钮总是会被自动调整到大于需要弹出的 popup 的宽度（ popup-menu 没有这个功能）， 如果弹子节点的某个选项很长的话那么按钮就会不必要的显示的太长，这时候可以使用类似下面的 CSS 让按钮自适应标题的长度：

CSSS!
```
.my-popup-selector{
    overflow:hidden;
    white-space:nowrap;
    width:calc( text-width( self.child(1):value ) + 20px ) !important;
}
```

关于 CSS 中 calc 函数的用法可参见 CSSS!脚本

HTMLayout 另外一个 behavior: dropdown-select 也可以实现下拉列表框，与 popup-selector 的区别是：popup-selector 是一个单纯的弹出列表控件 - 弹出节点的是一个弹出菜单，并且该菜单是可以多个控件共享的。而 dropdown-select 弹出节点不是一个弹出菜单并且也不能多控件共享，behavior: dropdown-select 的功能更多，可以添加 editable 支持编辑框，也可以对子节点进行分组、子节点也可以是一个树视图(treeview)。

在 HTMLayout 里，select 控件可以显示为下拉列表框，behavior 为 select-dropdown，代码如下：

```
<select>size 属性默认等于 1</select> 或者
<widget type="select-dropdown">...</widget>
```

注意还有一个 behavior: popup-selector 也可以实现下拉列表框，区别是 popup-selector 是弹出一个菜单，并且多个 popup-selector 可以共用同一个弹出菜单。

select 控件还可以显示普通的展开的列表框，类似 listbox 控件，这时候 behavior 为 select，HTML 代码如下：

```
<select size=N>N 指定显示行数，必须大于 1</select>
<widget type="select">默认显示为列表框（也没有下拉按钮，直接显示列表）</widget>
```

30.listview 实现列表视图

在 HTML 里，使用普通的 table 表格就可以显示出列表视图(listview)的效果，但是这样并不能具备 listview 的交互行为，例如按方向箭在列表项间移动等等。

这时候可以使用标准库中提供的 behavior:grid; 需要事先在 aardio 代码中调用 import web.layout.behavior.grid; 导入该模块。
然后在表格的 CSS 属性中指定 behavior:grid;就可以显示为列表视图了， grid 实现 listview 的一个主要优势是可以实现多选 (默认支持按 CTRL,SHIFT 组合键多选，可以响应 onTableRowClick 等事件, 关于 grid 请参考 aardio 里自带的 HTMLayout 范例、以及标准库的源码这里暂不多讲。

在 HTMLayout 里，select 控件默认显示为下拉列表框，代码如下:

```
<select>size 属性默认等于 1</select>
或者
<widget type="select-dropdown">...</widget>
//select 控件默认加载的 behavior 为"select-dropdown" （下拉框控件），而不是"behavior:select"。
```

behavior:select-dropdown 类似 behavior: popup-selector 都是用来实现下拉框控件。
而"behavior:select" 则主要是用来显示一个展开的列表框(listbox) 、或者列表视图(listview)、也可以用来实现树形视图。

下面的 HTML 代码会默认加载"behavior:select" ，显示为列表框（没有下拉按钮，直接显示列表）

```
<select size=N>...</select>
// N 是大于 1 的数，指定显示行数，显示 1 行时为下拉列表，显示多行列表框
<select type=“select”>...</select>
// 注意这里即使 size=1 也会显示为 select，而不是下拉框（select-dropdown）
<widget type=“select”></widget>
```

举个例子，下面的 HTML 代码显示为一个简单的列表框(listbox) :

```
<widget type="select">
    <OPTION VALUE=0 SELECTED>列表项 1</OPTION><OPTION VALUE=1 >列表项 2</OPTION>
</widget>
```

当然比起 winform 中传统的 listbox 控件 - 他可以实现更多复杂的显示效果，OPTION 节点里还可以写其他的 HTML。

HTMLayout 还可以在 CSS 中使用 flow 属性轻松的改变节点的布局、排列方式

```
<style>
    widget[name=iconview] {
        flow:h-flow;
        width:500px;
    }
    widget[name=iconview] OPTION:nth-child(3n+1)
    {
            clear:left; /*每行显示 3 列 */
    }
</style>
```

```
<widget type="select" name="iconview">
    <OPTION VALUE=0 SELECTED>列表项 1</OPTION>
    <OPTION VALUE=1 >列表项 2</OPTION>
    <OPTION VALUE=1 >列表项 4</OPTION>
    <OPTION VALUE=1 >列表项 4</OPTION>
    <OPTION VALUE=1 >列表项 5</OPTION>
    <OPTION VALUE=1 >列表项 6</OPTION>
</widget>
```

这类似于传统窗口控件中的 listview 控件以图标(icon)模式显示的效果。

列表框中的内容可以是一个表格(table) ，每行可以显示多列，实现类似报表视图的效果，表格的的每行(tr) 节点指定 role="option" 属性表示列表中的一行，HTML 简单示例如下:

```
<widget type="select" >
    <table>
        <tr><th>标题一</th><th>标题二</th></tr>
        <tr role="option" value=3> <td>第一行第一列</td>  <td>第一行第二列</td> </tr>
        <tr role="option" value=4> <td>第二行第一列</td>  <td>第二行第二列</td> </tr>
        <tr role="option" value=5> <td>测试</td> <td>测试</td> </tr>
        <tr role="option" value=6> <td>测试</td> <td>测试</td> </tr>
    </table>
</widget>
```

下面是一个完整的 aardio 示例:

效果图


```
import win.ui;
/*DSG{{*/
var winform = ..win.form( bottom=399;parent=...;right=599;text="HTMLayout 列表框" )
winform.add(  )
/*}}*/

import web.layout;
var wbLayout = web.layout(winform)

wbLayout.html = /**
```

```
<HTML>
    <HEAD>
    <STYLE>
    body { font:10pt verdana; }

    #report
    {
      width:*;
      height:*;
    }
    #report th
    {
      background-color: threedhighlight threedhighlight threedlightshadow threedlightshadow;
      border:1px solid threedlightshadow;
      padding:4px;
    }
    #report td
    {
      context-menu: selector(menu#menuId); /*selector 函数的参数是指定右键菜单的 CSS 选择器*/
      border:1px solid threedlightshadow;
      padding:4px;
    }
    #report tr:nth-child(even) /* every even row */
    {
      background-color: mintcream;
    }

    #report tr:current td /* current row */
    {
      background-color: highlight;
      color: highlighttext;
    }
    </STYLE>
    </HEAD>
    <BODY>

    <widget type="select" id="report"  >
        <table border=1 width=100%% cellspacing=-1px> <!-- cell border collapsing hack-->
        <tr><th>标题字段</th><th>值</th></tr>
        <tr role="option" value=1> <td>测试</td> <td> <INPUT type="checkbox" name="cb" />复选框 </td>
</tr>
        <tr role="option" value=2> <td>测试</td> <td> <INPUT name="Age" type="number" value="1" size="6
" step=1 minvalue=-9 maxvalue=100> </td> </tr>
        <tr role="option" value=3> <td>测试</td>  <td>测试</td> </tr>
        <tr role="option" value=4> <td>测试</td>  <td>测试</td> </tr>
        <tr role="option" value=5> <td>测试</td> <td>测试</td> </tr>
        <tr role="option" value=6> <td>测试</td> <td>测试</td> </tr>
```

```
        </table>
    </widget>

    <menu.context id="menuId">
                <li id="i1">菜单项</li>
                <li id="i2">菜单项</li>
                <hr/> <!-- 分隔线 -->
                <li>子菜单
                    <menu>
                    <li id="i5">子菜单项</li>
                    <li id="i6">子菜单项</li>
                    </menu>
                </li>
        </menu>

    </BODY>
</HTML>
**/

//改变选项时触发此事件
wbLayout.onSelectSelectionChanged = {
    report = function (ltTarget,ltEle,reason,behaviorParams) {
        winform.text =  ltTarget.value ;//ltTarget.value 获取被选中行的值
    }
}

//右键菜单触发下面的函数
wbLayout.onMenuItemClick = function (ltTarget,ltEle,reason,behaviorParams) {
    var elOption = wbLayout.queryEle("#report tr:current");
     winform.msgbox("你点击了菜单 " + ltTarget.innerText + ' \n 当前选中行:' + elOption.outerHTML )
}

winform.show()
win.loopMessage();
```

为列表项添加右键菜单

在 HTML 源码中</BODY>前面添加下面的代码

```
<menu.context id="menuId">
        <li id="i1">菜单项</li>
        <li id="i2">菜单项</li>
        <hr/> <!-- 分隔线 -->
        <li>子菜单
            <menu>
            <li id="i5">子菜单项</li>
            <li id="i6">子菜单项</li>
            </menu>
```

```
        </li>
</menu>
```

然后修改 CSS 中表格样式，使用 context-menu 属性为表格指定右键菜单，如下：

```
#report td
 {
    context-menu: selector(menu#menuId); /*selector 函数的参数是指定右键菜单的 CSS 选择器*/
    border:1px solid threedlightshadow;
    padding:4px;
 }
```

最后添加下面的 aardio 代码以响应菜单事件：

```
//右键菜单触发下面的函数
wbLayout.onMenuItemClick = function (ltTarget,ltEle,reason,behaviorParams) {
     var elOption = wbLayout.queryEle("#report tr:current");
      winform.msgbox("你点击了菜单 " + ltTarget.innerText + ' \n 当前选中行:' + elOption.outerHTML )
}
```

其他一些仅用于显示特效的 behaivior 请参考 HTMLayout 自带范例：
behavior:marquee 走马灯效果
behavior:reflection 水中倒影效果
behavior:magnifier 鱼眼特效

四、CSSS!脚本

1. 概述

CSSS! 是在标准 CSS 语法基础上扩展的一种简单脚本语言，以实现一些简单的交互行为。
CSSS! 基于标准 CSS 语法，通常是以一个属性名称（该名称以惊叹号结束)标明触发的事件，以逗号分隔语句（可不是一般编程语言中使用的分号哦），而以分号结束语句块（不是大括号哦），这些非常规的规则为是了遵守 CSS 语法规范。

2. 入门

让我们看看下面这段 CSSS! 脚本:

```
.item {
        hover-on! :
                ele = $1( input.url ) ,
                ele:empty == true ?
                        (self.value = "empty") #
                        (self.value = "filled"),
                ele:hover = true,
                self::width = ele.box-content-width(),
                self.$(.icon) -> @(ele) ele::background-color = rgb(255,0,0)
        ;
}
```

它的格式看起来就像是个扩展的 CSS 属性一样。

一些要注意的地方:

- CSSS! 语句使用逗号","作为语句结束符。
- CSSS! 中的字符串只能双引号标识，不能使用单引号("string")。
- CSSS! 中使用关键字 self 表示当前对象。

3. 触发事件

在前面的示例中，hover-on! 是我们的脚本要处理的事件标识。
当具有 .item 类的元素被鼠标悬停时，会触发此事件并执行里面的代码。
下面是完整的事件支持列表:

hover-on!:	鼠标悬停在节点上触发，类似 aardio 中的 wbLayout.onMouseEnter 事件
hover-off!:	鼠标离开节点时触发，类似 aardio 中的 wbLayout.onMouseLeave 事件
active-on!:	鼠标按下时触发，类似 aardio 中的 wbLayout.onMouseDown 事件
active-off!:	鼠标抬起时触发，类似 aardio 中的 wbLayout.onMouseUp 事件
focus-on!:	获得输入焦点时触发
focus-off!:	失去输入焦点时触发
key-on!: key-off!:	键盘的按键按下/抬起时触发，通过 key-code() 函数获取按键信息。 key-code() 获得的按键信息可能是一个用单引号包含的有效字符('a', '4', '$')或是下列预定义值之一: 'RETURN', 'LEFT', 'RIGHT', 'UP', 'DOWN', 'PRIOR', 'NEXT', 'TAB', 'HOME', 'END', 'DELETE', 'INSERT', 'BACK'
click!: double-click!:	鼠标单击，或双击时触发，必须绑定了所有允许接收单击，或双击事件的 behavior，储如 button ,hyperlink 等等。其他节点一般不会触发这两个事件（这些节点也可通过在 CSS 中指定 behavior:clickable; 启用该事件） 。
animation-start!: animation-step!: animation-end!:	动画控制事件 animation-start! 在调用了元素的 ele.start-animation() 方法后触发。 animation-step! 事件处理的最后必须返回一个整数(下次执行的间隔毫秒数)。 例如: return 500; 如果 animation-step! 返回的是 return cancel; 则动画中止, 同时触发 animation-end! 事件。
timer!:	定时触发器，配合 start-timer(ms) 和 stop-timer() 函数使用.start-timer(ms) 中的参数单位为毫秒。
size-changed!:	元素大小发生改变时触发
value-changed!:	input 类控件的值发生变时时触发

assigned!:	所属的样式被应用到元素上时触发
	写在这后面的脚本作为 CSS 属性被应用到节点时触发，而不是等待用户的什么交互。这个事件的作用类似于在 behavior 里的 onAttach:
	onAttach = function(ltEle){
	  return true
	}

validate!:	表单提交前的数据验证事件

我们下面看一个简单的 aardio 范例
假设有一个节点内容是这样:

HTML	<label for="id">

下面的 CSS 选择器 [for] 匹配所有指定了 for 属性的节点，在 CSS 选择器中，中括号指定 HTML 属性

CSS	[for] {

```
    hover-on! : $1(#< self.for >):hover = true;
    hover-off! : $1(#< self.for >):hover = false;
    active-on! : $1(#< self.for >):focus = true;
    cursor:pointer;
}
    input:hover {outline: 4px glow blue 1px;}
```

鼠标悬停或离开节点导致节点 for 属性绑定的节点同步改变 :hover 状态，而点击 lable 节点，for 属性绑定的节点将获得输入焦点，下面我们看一下 aardio 实现的完整范例：

```
import win.ui;
/*DSG{{*/
var winform = ..win.form( bottom=399;parent=...;text="aardio Form";right=599 )
winform.add(  )
/*}}*/

import web.layout;
wbLayout = web.layout(winform)

wbLayout.html = /**
<input type="text" />

<input type="text" id="myText" value="" />
<label for="myText" >鼠标放到这里，并点击这里</lable>
**/

wbLayout.css = /**
[for]
    {
        hover-on!: $1(#< self.for >):hover = true;
        hover-off!: $1(#< self.for >):hover = false;
        active-on!: $1(#< self.for >):focus = true ;
        cursor:pointer;
    }
    input:hover {
        outline: 4px glow blue 1px;
    }
**/

winform.show()
win.loopMessage();
```

4. 语法

CSSS! 嵌入标准 CSS 并遵守 CSS 语法规则,受限于 CSS 语法，脚本并不使用分号分隔语句，也不会使用大括号表示语句块，而是使用逗号分隔语句，使用分号结束语法块。

4.1 标志识

标志符可以使用英文字母, '-', '_' 或者 '$'符号,在这些后面可以附加数字，注意 CSS 允许横线作为标志符的一部分，这与其他编程语言不同，如果你需要用横线表示操作符，例如负号，那么清添加空格以区别。

4.2 关键字

true	逻辑真	false	逻辑否	null	空值	self	自身
这里的 self 可不是 aardio 中表示当前名字空间的 self 对象，实际上他更象 aardio 中表示自身的 this 或者 owner 对象。在 CSSS!脚本中， self 指当前触发事件的节点（类似 javascript 中的 this 对象）							

4.3 操作符

<	>	<=	==	!=	>=	&&	||+	-	*	/	%	^	|	&				

4.4 三元操作符

这个语法的作用类似条件判断语句,也有点象 aardio 中的三元操作符（区别是在 CSSS!中使用#号，而不是类似 aardio 或 C++中的冒号，当然原因也很简单，冒号被 CSS 语法占用了）

self:value > 12 ?(self:current = true, self.scroll-to-view());
上面这句代码是指，如果当前节点的值大于 12，那么选中当前节点(改变 current 状态)，并且滚动到视图范围。

4.5 类据类型

null	空值
boolean	逻辑值，两个可选值 true 或 false
integer	32 位整数值
float	64bit 浮点值(实际上这里指的是 aardio 或 C++中的 double 类型)
string	UNICODE 字符串
object	DOM 节点对象(引用)
object-set	DOM 节点对象的集合，函数 $() 返回这样的类型
function-reference	CSSS!中定义的函数

4.6 字面值

支持整数，浮点小数，文本，长度单位，以及正则表达式等字面量 ：

整数格式	[0-9]+ | '0x' [0-9A-Fa-f] |+ | '''字符''' |+键代码
浮点数格式	[0-9]+ '.' [0-9]+
浮点数格式	[0-9]+ '.' 'e'|'E' '+'|'-' [0-9]+
文本字面量	'"' [.]* '"'
正则字面量	'/' <单行正则表达式> ['/ig'] ';'
长度字面量	<整数格式>|<浮点数格式>| 'pt' | 'px' | 'pc' | '%' | 'em' | 'ex' | 'in' | 'cm' | 'mm'

字面量使用示范:

34	表示普通整数
0xAFED1234	表示十六进制整数

1.52	表示浮点数
1.e2	表示浮点数
1.e-2	表示浮点数
"Hello world!"	表示字符串
12pt	表示 12 象素
'A'	表示 A 的键盘代码，实际是一个数值

4.7 键盘代码字面量

键盘代码放在单引号中，可以是单个英文字母，也可以是下面的值：

'RETURN' | 'LEFT' | 'RIGHT' | 'UP' | 'DOWN' | 'PRIOR' | 'NEXT' | 'TAB' | 'HOME' | 'END' | 'DELETE' | 'INSERT' | 'BACK'

如果按下组合键，例如 Ctrl+X 则在键名前加上 '^'。
键盘代码的示例：

key-on!: key-code() == '^A'? ... #	Ctrl+A 组合键
key-code() == '^NEXT'? ... ;	Ctrl+PgDown 组合键

4.8 注释语法

注释语句被忽略不会被编译，类似 aardio 的注释语法有两种格式:

单行注释	'//' <无回车换行文本>
多行注释	'/*' <文本> '*/'

4.9 语句

CSSS! 脚本使用逗号 ','分隔语句，语句块以分号 ';'.结束 ， 下面是支持的语法规则:

4.9.1 赋值语句

<左侧表达式> '=' <表达式>

4.9.2 变量声明

<变量名>'=' <表达式>

4.9.3 语句块

语句块是一系列使用逗号分隔的语句，语法规则如下：

'(' <表达式> [, <表达式>]* ')'

4.9.4 条件语句

<逻辑表达式> '?' <条件为真时表达式> ['#' <条件为假时表达式>]

CSSS! 的条件判断语句格式很简单,类似 Javascript 或 aardio 中 的 三元操作符 ... ? ... : ... 组合, 区别是用#号代替了冒号

4.9.5 循环

不支持循环，但是函数可以支持尾递归调用。

4.9.6 Return 语句

return 语句用在函数中返回一个值:

```
return <表达式或值>
```

4.9.7 For each，枚举语句

```
<元素集合> '->' <单参数函数>
```

枚举通常被用于处理元素集合，调用右侧的函数遍历左侧元素集合，像这类的伪函数有 $()、$p() 和 $c()，格式：集合 -> @(参数名) 操作语句

示例:

```
CSSS!  input.number {
           value-changed! :
              total = 0,
              //定义一个变量 total 并且初始化他的值为 0

              $(input.number) -> @(el) total = total + el:value,
              /*定义一个函数 @(el) total = total + el:value，枚举所有匹配 CSS 选择器 input.number 的节点，并执行函数*/

              $1(td#total):value = total;
              //将 total 的值赋于第一个匹配 CSS 选择器 td#total 的节点
       }
```

5. 访问 DOM 节点的属性、状态值

5.1 可以使用成员操作符圆点访问节点的属性

例如对于当前节点: <input type="text"> 有下面的脚本代码:

```
t = self.type   //将会获取节点的 type 属性并赋值给变量 t
```

5.2 节点的状态使用冒号作为成员操作符

支持的 CSS 状态储如: :hover, :active, :link, :checked, :current, etc.

5.3 CSS 属性使用两个冒号'::' 作为成员操作符

```
self::opacity = self::opacity + 0.1 #
return cancel;
//在定时器中逐渐的将对象改变为透明状态。
```

5.4 DOM 元素对象

$1()函数返回一个对象，这个对象表示一个 DOM 元素。DOM 元素制成以下 3 种访问器:

- ele.attribute - 访问 DOM 元素的属性。
- ele:state-flag - 访问 DOM 元素的运行期标志: hover, active, focus 等。
- ele::style-attribute - 访问 DOM 元素的样式(CSS)属性。

DOM 元素也有许多方法，这些方法通过下面的语句访问: domobj.function-name()。

5.4.1 DOM 元素属性

DOM 元素可以设置或移除任何属性。
要移除一个元素的某个属性，只需设置这个属性值为 null:

```
self.show = true; // 将会设置 show 属性的值为"true";
self.show = null; // 将 self 元素的 show 属性移除。
```

5.4.2 DOM 元素的状态值

元素有合成状态和纯状态。下面的代码在链接被点击时会为链接设置:visited 状态。

```
a:link
{
   click!: self:visited = true;
}
```

合成状态字段:

状态	说明
ele:value	代表 DOM 元素的值。对于输入(input)元素(被附加的对应行为处理的元素)，这个值对应 value 值；对于纯 DOM 元素，是这个元素的文本内容。
ele:index	是元素在父容器中的索引。Index 是范围为 1 .. parent().children()间的一个整数值。

纯状态字段都是 boolean 类型的。这些状态标志的名称和对应的 CSS 伪类相同:

状态	说明
ele:link	任何用于 href 属性的元素
ele:hover	鼠标悬停处的元素
ele:active	被激活(按下)的元素
ele:empty	该元素是否是空的——即没有子元素和内容文本
ele:readonly	只读元素，行为相关标志
ele:disabled	被禁用的元素，行为相关标志
ele:focusable	可接受焦点的元素
ele:visited	辅助标志 - 目前内部不使用
ele:current	容器中的当前项，如<select>中的当前的<option>
ele:checked	被勾选(选择)的元素，如复选框或多选列表
ele:expanded	处于展开状态的元素 - 如树视图中的节点、选择列表中<options>
ele:collapsed	与 ele:expanded 状态相反
ele:incomplete	元素有图像(back/fore/bullet)请求，但是未交付
ele:animating	当做正处于动画中
ele:anchor	<select miltiple>中的第一个元素，ele:CURRENT 是当前元素。
ele:synthetic	合成 DOM 元素 - 如表格中所有缺失的单元格(<td>)将会设置这个标志

ele:owns-popup	可见弹出框的锚点(拥有者)元素
ele:tab-focus	通过 tab 键获取焦点的元素。设置该标志的同时会设置:focus 标志
ele:busy	元素是忙的。如果元素使用 HTMLayoutRequestElementData 请求额外数据时会设置这个标志，当请求数据到达后会重置这个标志。
ele:drag_over	拖拽时经过的可以接受拖拽源的块元素(所以是当前拖拽目标)。这个标志设置给拖拽目标块。在任何时刻只会有一个这样的块元素。
ele:drop-target	当前拖拽目标元素，当 D&D 处于激活状态时，多个元素可以有这个状态标志。
ele:moving	dragging/moving - 这个标志设置在正在移动的元素(拖拽源的副本)。
ele:copying	dragging/copying - 这个标志设置在正在拷贝的元素(拖拽源的副本)。
ele:drag-source	被拖拽的元素
ele:popup	处于弹出状态的元素
ele:pressed	按下 - 于 actvie 状态相近，不过有更长的生命区间 - 在 MOUSE_UP 事件之前这个状态一直存在，所以行为中可以在 MOUSE_UP 事件中检查这个状态来发现点击条件。
ele:has-children	有一个活多个子元素
ele:has-child	只有一个子元素
ele:ltr	该元素或它的最近父容器有@dir，并且 dir 值为"ltr"
ele:rtl	该元素或它的最近父容器有@dir，并且 dir 值为"rtl"

5.4.3 DOM 元素的样式属性

CSSS!允许设置所有支持的 CSS 属性值。支持的 CSS 属性列表可以在 支持的 css 样式 找到。
下面的样式规则使 div 元素在获得 fade-out 属性时显示 fade-out 特效:

```
div[fade-out]
{
  assigned!: self::opacity = 1.0, self.start-timer(50);
  timer!: self::opacity < 1.0?
            self::opacity = self::opacity + 0.1 # return cancel /* 停止计时器 */;
}
```

5.4.4 DOM 元素支持的函数(方法)

下面是 CSSS!中为 DOM 元素定义的方法集合:

ele.parent()		DOM 元素, 返回当前元素的父元素
ele.child(n:integer)		获取第 n 个子元素. 参数范围 1 ... ele.children()
ele.children()		integer, 返回元素的子元素数量
ele.child(n:integer)		DOM 元素, 返回在 *n* 位置的子元素。*n* 是一个范围在 1..children()间的正数
ele.next()		DOM 元素, 返回当前元素的下一个兄弟元素
ele.previous()		DOM 元素, 返回当前元素的上一个兄弟元素
选	$1(.item)	获取匹配 ".item" 的第一个元素 类似 aardio 中的 wbLayout.queryEle()函数(唯一的区别是在 aardio 中需要将 CSS 选择器参数放在引号中 - 即使用字符串包含 CSS 选择器)
	$(.item)	获取所有匹配 ".item" 的元素 类似 aardio 中的 wbLayout.queryEles()函数(在 aardio 中需要将 CSS 选择器参数放在引号中 - 即使用字符串包含 CSS 选择器) $(.item) 返回的集合可以象单个元素一样直接使用,例如:

择器方法		$(.item).属性 = 值, 返回值也可以用于枚举, 例如: $(.item)->ele(ele) ele.属性 = 值;
	ele.$(item)	返回满足选择条件的所有元素集合 这个返回只会检索 ele 的子元素 类似 aardio 中的 ele.queryElements()函数 item 中的:root 伪类匹配 ele 元素 示例: self.$(:root > li):expanded = true 将仅会为 self 的直接 li 子元素设置 expanded 状态
	ele.$1(item)	同上，区别是仅返回一个满足选择器的第一个元素
	ele.$p(item)	返回匹配 item 的 ele 元素的父元素集 item 中的:root 是全局的 DOM 根元素
	ele.$1p(item)	返回父元素匹配 item 最近的一个元素 类似 aardio 中的 ele.queryParent()函数 item 中的:root 是全局的 DOM 根元素
ele.start-timer(period:integer)		在 ele 元素上启动一个时间间隔为 period(毫秒)的计时器 这个计时器在 period 间隔后将会触发 ele 元素的 timer!事件 要停止计时器，可以调用 stop-timer()函数或在 timer!事件中 return cancel
ele.stop-timer()		停止先前使用 start-timer()方法启动的计时器。
ele.scroll-to-view()		确保 ele 元素是可见的，滚动节点到当前视图
ele.box-type-what()		是一个方法集合，它们可以检测 ele 的度量尺寸 示例: self.box-border-width() 将会返回元素的边框盒的宽度(像素) 函数中的 type 部分可以是下面值之一: margin - 元素的外边距空间; border - 元素的内边框空间; padding - 元素的内边距空间; content - 元素的内容空间(内容轮廓盒); inner - 元素的内部空间; client - 元素的客户空间 - 元素的内部空间减去滚动条区域，如果没有滚动条则返回内部空间 what 部分的名称可以是以下值之一: left - 盒的左边距; right - 盒的右边距; top - 盒的上边距; bottom - 盒的下边距; width - 盒的宽度距; height - 盒的高度距; 这些名称组合起来共有 30 个 ele.box-...-...()函数
ele.x-what()		一个函数集合，它们检测 ele 的相对 x 位置 函数名中的 what 部分可以有以下值: parent - 元素相对于它的父元素的水平坐标; root - 元素相对于根元素(<html>)的水平坐标; view - 元素相对于视图(窗口)的水平坐标;

	screen - 元素相对于屏幕的水平坐标;
ele.y-what()	一个函数集合，它们检测 ele 的相对 y 位置 函数名中的 what 部分与上一个函数相同
ele.system-scrollbar-width()	系统设定的滚动条宽度,高度
ele.system-scrollbar-height()	
ele.start-animation([duration])	启动 animation-start!、animation-tick!、animation-end!事件中定义的动画 duration 参数可以接受秒值: self.start-animation(0.4s) - 将会启动持续 400ms 的动画
ele.stop-animation()	停止执行动画。最好不要同时启动多个动画效果，停止当前动画再启动效的动画
ele.text-width("string")	返回当前样式下的 ele 元素的 string 字符串的像素宽度
ele.min-intrinsic-width()	节点最小,最大内容宽度
ele.max-intrinsic-width()	
ele.min-intrinsic-height()	节点最小,最大内容高度
ele.max-intrinsic-height()	
ele.system-border-width()	系统设定的边框宽度
ele.system-small-icon-width()	系统设定的小图标宽度,高度
ele.system-small-icon-height()	
ele.foreground-image-width()	当前节点的前景图片高度,高度
ele.foreground-image-height()	
ele.background-image-width()	当前节点的背景图片高度,宽度
ele.background-image-height()	
ele.refresh()	该方法将会导致 ele 的尺寸重新计算。如果 width 或 height 使用 calc()修改后，需要调用这个方法

5.4.5 DOM 元素的事件

CSSS!扩展了经典 CSS，通过下面的这些事件属性，可以定义事件处理器，参见 触发事件

5.5 Self 对象

self 关键字指代规则应用的 DOM 元素对象，在脚本事件中表示当前节点。

5.6 Cancel

这个关键字是专用于 return 语句，以终止事件继续传递。

5.7 函数

CSSS! 支持匿名函数定义

5.7.1 定义函数

'@(' <参数列表> ')' <语句> | <语句块>

其中，<参数列表>是一个以逗号分隔的名称列表。
函数可以赋值给变量:

```
foo = @( a, b ) c = a + b, return c

类似于 aardio 中类似的定义函数的代码如下:
foo = function( a, b ) {
    c = a + b; return c;
}
```

```
foo = @(p1, p2) p1 + p2, ...
foo = @(p1, p2) (p1 + p2, ...), ...
//函数声明 p1 和 p2 两个参数，函数体 p1 + p2 或(p1 + p2, ...)，将函数引用赋值给 foo 变量。
```

5.7.2调用函数

```
<变量> '(' <参数-1> , ... <参数-n> ')'
```

示例，下面的代码将会调用上面的 foo 函数，并且将函数的返回值设置到 bar 变量:

```
bar = foo(1,2)
```

5.7.3 常用函数

5.7.3.1 CSS 选择器函数

见 DOM 元素支持的函数(方法) 表部分函数:
$1(.item)、$(.item)、ele.$1(.item)、ele.$(.item)、ele.$1p(.item)、ele.$p(.item)、ele.parent()、ele.next()、ele.previous()、ele.child(n:integer)、ele.children()

CSS 选择器参数中可以使用变量(包含在尖括号中)，示例:

```
ncol = 2, $( table td:nth-child(< ncol >)).some-attr = "hi!";
//对所有符合 CSS 选择器 table td:nth-child(2) 规则的节点改变属性 some-attr 的值为 "hi!"
```

5.7.3.2 calc()函数

在 calc(<表达式>)你可以使用下面的语句访问 DOM 属性和元素的状态:

self.attribute	访问元素的 DOM 属性
self:state-flag	访问 DOM 元素的运行期状态标志: hover、active、focus 等

其中，self 是调用 calc()函数的元素引用。calc 是 CSS3 中的一个用于在运行时计算属性值的函数，
函数的参数里可以使用表达式，支持 + - * /等算术运行，例如在 CSS 中指定高度属性可以这样写:

CSS	div.cls{
	height:calc(100% - 5px);
	}

要注意 + - * /两边一定要有空格，不能写为 calc(100%-5px); 在标准的 content-box 盒模型下,如果宽度为 100% - 且指定边框边距等可能就会撑破页面，这时候就可以使用 calc 函数在运行时计算出自适应的合适宽度。
在 HTMLayout 中 calc 还能象 CSSS!脚本那样访问 DOM 节点对象的方法和属性。参见 DOM 元素支持的函数(方法) 表部分函数:
ele.children()、ele.child(n:integer)、ele.next() 、ele.previous() 、ele.parent() 、ele.text-width("string") 、ele.min-intrinsic-width(), ele.max-intrinsic-width()、ele.min-intrinsic-height()、ele.max-intrinsic-height() 、ele.system-scrollbar-width

()、ele.system-scrollbar-height()、ele.system-border-width()、ele.system-small-icon-width()、ele.system-small-icon-height() 、ele.foreground-image-width(),ele.foreground-image-height()、ele.background-image-width(),ele.background-image-height()

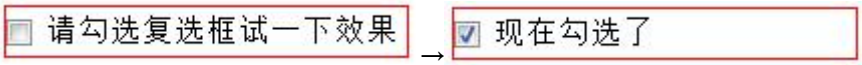
CSS 选择器函数:
ele.$(item)、ele.$1(item)、ele.$p(item) 、ele.$1p(item)

示例:

CSSS!	div {
	width:calc(min-intrinsic-width());
	}

你可能会说，直接写 div { width:min-intrinsic } 不是更简单么？！
calc() 在括号里面不仅仅是可以写简单的函数调用，还支持一些更复杂的表达式，请看一个完整的示例:

效果图




```
<html #test>
<head>
  <style>
        button.with-text {
            border:1px solid red;/*红色边框*/
            width: calc(
                        max( text-width( self.child(1):value ),text-width( self.child(2):value ) )
            );
        }
        button.with-text:checked > text:nth-child(1) {
                    display:none;
        }
        button.with-text:not(:checked) > text:nth-child(2) {
                    display:none;
        }
  </style>
</head>
<body>
  <button .with-text type="checkbox">
    <text>请勾选复选框试一下效果 </text>
    <text>现在勾选了</text>
  </button>
</body>
</html>
```

注意上面源码中这段 CSS 代码:

```
width: calc(
    max( text-width( self.child(1):value ),text-width( self.child(2):value ) )
);
```

首先使用 self.child(1):value, self.child(2):value 分别取到这两个节点的文本值,然后使用 text-width(文本值) 分别计算出这两段文本的显示宽度。最后使用 max(文本 1 宽度,文本 2 宽度) 取最大值返回。

calc()表达式仅仅是 CSSS!语法的一个子集，并不能支持所有的 CSSS!语法 (例如 a ? b # c 即不支持)
因为 CSSS!里的 assigned!事件在 CSS 应用到节点时会立即执行，所以上面的 calc() 表达式也可以用等价的 CSSS!代码来写，如下:

```
button.with-text {
    assigned!: self::width =
        max( self.text-width( self.child(1):value ),self.text-width( self.child(2):value ) )
    ;
}
```

所以 calc() 实际上是一种缩写的 CSSS!的表达式。

5.7.3.3 morph() 变形动画函数

在 CSSS!脚本中，使用 ltEle.start-animation() 启动动画以后，在执行动画的 animation-step! 事件内，可选使用 morph() 函数来实现动画变形效果。
语法:

```
morph("ease-function-name", start-value, end-value )
```

ease-function-name 指定一种动画效果的名称，而每一种动画效果，都会从一个指定的 CSS 初始属性值 start-value 过渡变化到 en-value;
可选的 ease-function-name 有以下几种 (*****代替 bounce、back、elastic、circ、expo、sine、quint、quart、cubic、quad):
*****-in、*****-out、*****-in-out //共计 24 种

示例:

```
import win.ui;
/*DSG{{*/
winform = ..win.form( text="HTMLayout 变形动画函数 morph() 演示";bottom=399;parent=...;right=599;border="resizable" )
winform.add(
layoutWindow={ dr=1;dl=1;notify=1;right=580;left=10;dt=1;top=12;z=1;db=1;bottom=382;multiline=1;cls="edit" }
)
/*}}*/

import web.layout;
wbLayout = web.layout(winform.layoutWindow);

wbLayout.html =/***
<div id="myDiv" >请用鼠标点击这里</div>
***/

wbLayout.css = /**
#myDiv{
  width:200px;
  border:1px solid #DDD;
```

```
  background:#EEE;
  margin:10px;
  padding:5px;
  font:system;

  //鼠标点击事件
  active-on!:self.start-animation( 0.4s );

  //动画事件
  animation-step!:
    self::height = morph("cubic-out" //指定动画函数名
      , self.min-intrinsic-height() //指定变形初始值
      , 300px //指定变形完成后的值
    );
}
**/

winform.show()
win.loopMessage();
```

CSS 中的 transition 也能实现类似过渡效果，用法更简洁一些
参考: aardio 范例->HTMLayout->界面预览工具->transition 分类

一个简单例子

```
{
  width:100px;
  height:100px;
}
div:hover
{
  width:200px;
  height:200px;
  transition: width(linear,400ms)
              height(linear,400ms);
}
```

在 div:hover 状态中使用 transition 定义了变换效果，那么从其他状态切换到 hover 状态时将会应用该变换效果。
transition 可以指定一个名多个值，多个值使用空格分格，每个值的格式如下：

CSS 属性名(状态变换函数, 动画持续时长, 动画间隔延时)
CSS 属性名(渐入状态变换函数, 渐入动画持续时长, 渐入动画间隔延时, 渐出状态变换函数, 渐出动画持续时长, 渐出动画间隔延时)
CSS 属性名(渐入该状态变换函数, 渐入动画持续时长,渐入动画间隔延时, none)
CSS 属性名(none, 渐出该状态变换函数, 渐出动画持续时长,渐出动画间隔延时)

上面的间隔延时参数都可以省略

范例：

```
import win.ui;
```

```
/*DSG{{*/
var winform = ..win.form(text="transition 变换效果";right=599;bottom=399)
winform.add()
/*}}*/

import web.layout;
var wbLayout = web.layout( winform )

wbLayout.html = /**
<html>
<head>
  <style>

    div.container
    {
      flow:horizontal-flow;
      width:*;
      height:*;
    }
    div.cell
    {
      width:*;
      overflow:hidden;
    }
    div.cell:nth-child(4n)
    {
      clear:left; /* four in the row; */
    }

    div.test
    {
      background-color: gold;
      color: black;
      width:100px;
      padding:4px;
      margin:4px;
      white-space:nowrap;
      overflow:hidden;

    }
    div.test:hover
    {
      background-color: red;
      width:200px;
    }

    div.test:hover[ease="linear"]  { transition: width(linear,0.5s) background-color(linear,0.5s,0.5s); }
```

```
    div.test:hover[ease="quad-in"]          { transition: width(quad-in,0.5s)       background-color(quad-in,0.5s,0.5s); }
    div.test:hover[ease="quad-out"]         { transition: width(quad-out,0.5s)      background-color(quad-out,0.5s,0.5
s); }
    div.test:hover[ease="quad-in-out"]      { transition: width(quad-in-out,0.5s) background-color(quad-in-out,0.5s,0.
5s); }
    div.test:hover[ease="cubic-in"]         { transition: width(cubic-in,0.5s)      background-color(cubic-in,0.5s,0.5s); }
    div.test:hover[ease="cubic-out"]        { transition: width(cubic-out,0.5s)     background-color(cubic-out,0.5s,0.5s);
 }
    div.test:hover[ease="cubic-in-out"]     { transition: width(cubic-in-out,0.5s)background-color(cubic-in-out,0.5s,0.5
s); }
    div.test:hover[ease="quart-in"]         { transition: width(quart-in,0.5s)      background-color(quart-in,0.5s,0.5s); }
    div.test:hover[ease="quart-out"]        { transition: width(quart-out,0.5s) background-color(quart-out,0.5s,0.5s); }
    div.test:hover[ease="quart-in-out"]     { transition: width(quart-in-out,0.5s) background-color(quart-in-out,0.5s,0.
5s); }
    div.test:hover[ease="quint-in"]         { transition: width(quint-in,0.5s) background-color(quint-in,0.5s,0.5s); }
    div.test:hover[ease="quint-out"]        { transition: width(quint-out,0.5s) background-color(quint-out,0.5s,0.5s); }
    div.test:hover[ease="quint-in-out"]     { transition: width(quint-in-out,0.5s) background-color(quint-in-out,0.5s,0.5
s); }
    div.test:hover[ease="sine-in"]          { transition: width(sine-in,0.5s) background-color(sine-in,0.5s,0.5s); }
    div.test:hover[ease="sine-out"]         { transition: width(sine-out,0.5s) background-color(sine-out,0.5s,0.5s); }
    div.test:hover[ease="sine-in-out"]      { transition: width(sine-in-out,0.5s) background-color(sine-in-out,0.5s,0.5s);
 }
    div.test:hover[ease="expo-in"]          { transition: width(expo-in,0.5s) background-color(expo-in,0.5s,0.5s); }
    div.test:hover[ease="expo-out"]         { transition: width(expo-out,0.5s) background-color(expo-out,0.5s,0.5s); }
    div.test:hover[ease="expo-in-out"]      { transition: width(expo-in-out,0.5s) background-color(expo-in-out,0.5s,0.5
s); }
    div.test:hover[ease="circ-in"]          { transition: width(circ-in,0.5s) background-color(circ-in,0.5s,0.5s); }
    div.test:hover[ease="circ-out"]         { transition: width(circ-out,0.5s) background-color(circ-out,0.5s,0.5s); }
    div.test:hover[ease="circ-in-out"]      { transition: width(circ-in-out,0.5s) background-color(circ-in-out,0.5s,0.5s); }
    div.test:hover[ease="elastic-in"]       { transition: width(elastic-in,0.5s) background-color(elastic-in,0.5s,0.5s); }
    div.test:hover[ease="elastic-out"]      { transition: width(elastic-out,0.5s) background-color(elastic-out,0.5s,0.5s); }
    div.test:hover[ease="elastic-in-out"] { transition: width(elastic-in-out,0.5s) background-color(elastic-in-out,0.5s,0.
5s); }
    div.test:hover[ease="back-in"]          { transition: width(back-in,0.5s) background-color(back-in,0.5s,0.5s); }
    div.test:hover[ease="back-out"]         { transition: width(back-out,0.5s) background-color(back-out,0.5s,0.5s); }
    div.test:hover[ease="back-in-out"]      { transition: width(back-in-out,0.5s) background-color(back-in-out,0.5s,0.5
s); }
    div.test:hover[ease="bounce-in"]        { transition: width(bounce-in,0.5s) background-color(bounce-in,0.5s,0.5s);
 }
    div.test:hover[ease="bounce-out"]       { transition: width(bounce-out,0.5s) background-color(bounce-out,0.5s,0.
5s); }
    div.test:hover[ease="bounce-in-out"]  { transition: width(bounce-in-out ,0.5s) background-color(bounce-in-out
 ,0.5s,0.5s); }

  </style>
</head>
<body>
```

```
  <h1>transition 变换效果</h1>
  <div .container>
    <div .cell><div .test   ease="linear"           >linear</div></div>
    <div .cell><div .test   ease="quad-in"          >quad-in</div></div>
    <div .cell><div .test   ease="quad-out"         >quad-out</div></div>
    <div .cell><div .test   ease="quad-in-out"      >quad-in-out</div></div>
    <div .cell><div .test   ease="cubic-in"         >cubic-in</div></div>
    <div .cell><div .test   ease="cubic-out"        >cubic-out</div></div>
    <div .cell><div .test   ease="cubic-in-out"     >cubic-in-out</div></div>
    <div .cell><div .test   ease="quart-in"         >quart-in</div></div>
    <div .cell><div .test   ease="quart-out"        #test     >quart-out</div></div>
    <div .cell><div .test   ease="quart-in-out"     >quart-in-out</div></div>
    <div .cell><div .test   ease="quint-in"         >quint-in</div></div>
    <div .cell><div .test   ease="quint-out"        >quint-out</div></div>
    <div .cell><div .test   ease="quint-in-out"     >quint-in-out</div></div>
    <div .cell><div .test   ease="sine-in"          >sine-in</div></div>
    <div .cell><div .test   ease="sine-out"         >sine-out</div></div>
    <div .cell><div .test   ease="sine-in-out"      >sine-in-out</div></div>
    <div .cell><div .test   ease="expo-in"          >expo-in</div></div>
    <div .cell><div .test   ease="expo-out"         >expo-out</div></div>
    <div .cell><div .test   ease="expo-in-out"      >expo-in-out</div></div>
    <div .cell><div .test   ease="circ-in"          >circ-in</div></div>
    <div .cell><div .test   ease="circ-out"         >circ-out</div></div>
    <div .cell><div .test   ease="circ-in-out"      >circ-in-out</div></div>
    <div .cell><div .test   ease="elastic-in"       >elastic-in</div></div>
    <div .cell><div .test   ease="elastic-out"      >elastic-out</div></div>
    <div .cell><div .test   ease="elastic-in-out"   >elastic-in-out</div></div>
    <div .cell><div .test   ease="back-in"          >back-in</div></div>
    <div .cell><div .test   ease="back-out"         >back-out</div></div>
    <div .cell><div .test   ease="back-in-out"      >back-in-out</div></div>
    <div .cell><div .test   ease="bounce-in"        >bounce-in</div></div>
    <div .cell><div .test   ease="bounce-out"       >bounce-out</div></div>
    <div .cell><div .test   ease="bounce-in-out"  >bounce-in-out  </div></div>                    </div>
</body>
</html>
**/

winform.show()
win.loopMessage();
```

还可以使用 transition: blend; 用来实现渐入渐出的混合效果。

或者使用 transition:sound(in: url(sound1.wav) ,out: url(sound2.wav)) 用来指定切换状态的音响效果，其中 out 部分可省略。

5.7.3.4 其他全局函数

int(val)	转换一个值为整数
float(val)	转换一个值为浮点数，类似 aardio 中的 tonumber

length(val)	转换一个值为长度值（包含单位)
min(val1, val2 ... valN)	返回最小数
max(val1, val2 ... valN)	返回最大数
limit(val, minval, maxval)	返回一个值是并限定允许的最小或最大值

5.7.3.5 String 字符串对象的一些成员函数

String.length()	取字符串长度
string.toUpper()	将字符串转为大写
string.toLower()	将字符串转为小写
string.substr(start[, length = -1])	截取字符串，参数一指定开始位置，参数二指定长度，如果小于零，则截取到尾部

5.7.3.5 HTML 节点对象提供的一些内置函数

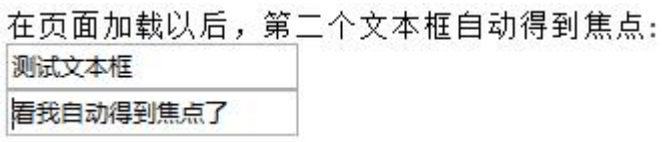
见 DOM 元素支持的函数(方法) 表部分函数：
ele.children() 、ele.child(n:integer)、ele.next() 、ele.previous() 、ele.parent() 、ele.text-width("string") 、ele.min-intrinsic-width(), ele.max-intrinsic-width() 、ele.min-intrinsic-height(), ele.max-intrinsic-height() 、ele.system-scrollbar-width(),ele.system-scrollbar-height() 、ele.system-border-width() 、ele.system-small-icon-width(),ele.system-small-icon-height() 、ele.foreground-image-width(),ele.foreground-image-height() 、ele.background-image-width(),ele.background-image-height()

6. CSSS!示例

已省略 arrdio 代码，只保留 html 代码

自动得到焦点

效果图




```
<html>
<head>
  <style>
  //下面的选择符表示拥有 autofocus 属性的节点，并且状态为可以获取焦点(focusable)
  [autofocus]:focusable {
    assigned!: self:focus = true; //assigned!事件表示加载脚本即时执行
  }
  </style>
<head>
<body>
  在页面加载以后，第二个文本框自动得到焦点:<br/>
  <input type="text" value="测试文本框" /><br/>
  <input type="text" value="看我自动得到焦点了" autofocus />
</body>
</html>
```

自动展开节点

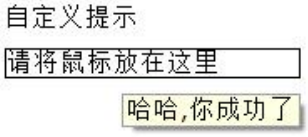
注意下面仅仅用到了 CSS 扩展功能,并没有直接使用 CSSS!脚本

效果图


```
<html>
<head>
  <style>
    div.slave { visibility:collapse; } //指定了 slave 类的节点默认折叠
    //注意下面的加号,这是一个关系选择符,表示加号前面的节点后面的第一个节点
    widget.master:checked + div.slave { visibility:visible; }
  </style>
<head>
<body>
  <widget type="checkbox" .master>请点选这里:</widget>
  <div .slave>
    <li>item #1</li>
    <li>item #2</li>
    <li>item #3</li>
  </div>
  <widget type="checkbox" .master>请点选这里试试:</widget>
  <div .slave>
    <li>item #1</li>
    <li>item #2</li>
    <li>item #3</li>
  </div>
</body>
</html>
```

自定义提示效果

效果图

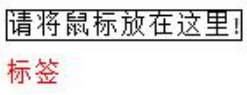



```
<html>
  <head>
    <style>
      p#test {
        border:1px solid black; //定义边框
        width: 30%; //定义宽度
        hover-on!: self.start-timer(400); //鼠标悬停时,开始定时器,间隔 400 毫秒
        hover-off!: self.stop-timer(),
```

```
                $1(popup#for-test):popup = false ; //注意冒号作为成员操作符时改变节点的状态
                //注意下面用到了条件语句,问号前面是条件
        timer!: self:hover ? self.show-popup($1(popup#for-test),3,9), /*参数表示在右下角弹出提示 */
                return cancel;//阻止事件继续传递
    }
    popup#for-test {
      margin-top:10px; /*指定顶边距 */
    }
  </style>
 </head>
<body>
 自定义提示
 <p #test>请将鼠标放在这里</p>
 <popup #for-test>
   哈哈,你成功了
 </popup>
</body>
</html>
```

改变标签颜色

效果图



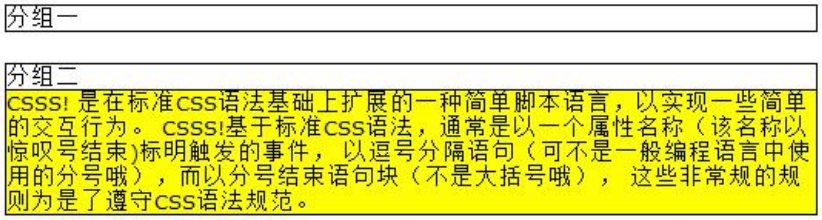

```
<html>
 <head>
  <style>
    a
    {
      hover-on!: $(p.slave):hover = true;
      hover-off!: $(p.slave):hover = false;
      border:1px solid;
    }
    p.slave:hover
    {
      color:red;
    }
  </style>
 </head>
<body>
 <a>请将鼠标放在这里!</a>
 <p .slave>标签</p>
</body>
</html>
```

非线性弹出动画

效果图

本示例实现思路：

- 通过改变self.min-intrinsic-height()的值来扩大元素的固有高度
- 使用 morph(ease-function-name, start-value, end-value) - 非线性缓冲函数




```
<html>
  <head>
    <style>
      div.panel
      {
        border:1px solid;
        margin:20px;
      }
      div.panel > caption
      {
        active-on!: panel = self.parent(),
                    panel.state = panel.state == "expanded" ? "collapsing" # "expanding";
      }
      div.panel > div.details
      {
        background:yellow;
      }
      div.panel[state="collapsed"] > div.details
      {
        overflow:hidden;
        height:0;
      }
      div.panel[state="collapsing"] > div.details
      {
        overflow:hidden;
        border-top:1px solid;
        assigned!              : self.start-animation( 0.4s );
        animation-step!: self::height   = morph("bounce-in", self.min-intrinsic-height(), 0px);
        animation-end! : self::height   = null, self.parent().state = "collapsed";
      }
      div.panel[state="expanding"] > div.details
      {
        overflow:hidden;
        height:0;
        border-top:1px solid;
        assigned! : self.start-animation(0.4s);
        animation-end!:    self::height = null, self.parent().state = "expanded";
        animation-step!:   self::height   = morph("bounce-in", 0px, self.min-intrinsic-height());
```

```
      }
      div.panel[state="expanded"] > div.details
      {
        border-top:1px solid;
      }
    </style>
  </head>
<body>
  <p>本示例实现思路：</p>
    <ul>
      <li>通过改变<code>self.min-intrinsic-height()的值来扩大元素的固有高度</code></li>
        <li>使用 <code>morph(ease-function-name, start-value, end-value)</code> - 非线性缓冲函数</li>
    </ul>
  <div .panel state="collapsed">
    <caption>分组一</caption>
    <div .details>CSSS! 是在标准 CSS 语法基础上扩展的一种简单脚本语言，以实现一些简单的交互行为。
        CSSS!基于标准 CSS 语法，通常是以一个属性名称（该名称以惊叹号结束)标明触发的事件，
        以逗号分隔语句（可不是一般编程语言中使用的分号哦），而以分号结束语句块（不是大括号哦），
        这些非常规的规则为是了遵守 CSS 语法规范。
    </div>
  </div>
  <div .panel state="expanded">
    <caption>分组二</caption>
    <div .details>CSSS! 是在标准 CSS 语法基础上扩展的一种简单脚本语言，以实现一些简单的交互行为。
        CSSS!基于标准 CSS 语法，通常是以一个属性名称（该名称以惊叹号结束)标明触发的事件，
        以逗号分隔语句（可不是一般编程语言中使用的分号哦），而以分号结束语句块（不是大括号哦），
        这些非常规的规则为是了遵守 CSS 语法规范。
    </div>
  </div>
</body>
</html>
```

非线性平滑动画

效果图

本示例实现思路：

- 通过改变self.min-intrinsic-height()的值来扩大元素的固有高度
- 使用morph(ease-function-name, start-value, end-value) - 非线性缓冲函数
- 使用self.start-animation(0.4s)在连续的时间内同时改变动画的多个属性

分组一

分组二

CSSS!基于标准CSS语法，通常是以一个属性名称（该名称以惊叹号结束）标明触发的事件，以逗号分隔语句（可不是一般编程语言中使用的分号哦），而以分号结束语句块（不是大括号哦），这些非常规的规则为是了遵守CSS语法规范。

```
<html>
  <head>
    <style>
```

```
    div.panel
    {
      border:1px solid;
      margin:20px;
    }
    div.panel > caption
    {
      active-on!: panel = self.parent(),
                  panel.state = panel.state == "expanded" ? "collapsing" # "expanding";
    }
    div.panel > div.details
    {
      background:yellow;
      overflow:hidden;
      height:min-intrinsic;
                -animation-ease:"cubic-out";
    }
    div.panel[state="collapsed"] > div.details
    {
      height:0;
    }
    div.panel[state="collapsing"] > div.details
    {
      border-top:1px solid;
      height:min-intrinsic;
      assigned!              : self.start-animation( 0.4s );
      animation-step!: ease-f = self::-animation-ease,
                         self::height  = morph(ease-f, self.min-intrinsic-height(), 0px),
                       self::opacity = morph(ease-f, 1.0, 0.0);
      animation-end! : self::height  = null, self::opacity = null, self.parent().state = "collapsed";
    }
    div.panel[state="expanding"] > div.details
    {
      height:0;
      border-top:1px solid;
      assigned! : self.start-animation(0.4s);
      animation-step!:  ease-f = self::-animation-ease,
                         self::height  = morph(ease-f, 0px, self.min-intrinsic-height()),
                          self::opacity = morph(ease-f, 0.0, 1.0);
      animation-end!:    self::height = null, self::opacity = null, self.parent().state = "expanded";
    }
    div.panel[state="expanded"] > div.details
    {
      border-top:1px solid;
    }
  </style>
</head>
```

```
<body>
  <p>本示例实现思路：</p>
    <ul>
      <li>通过改变<code>self.min-intrinsic-height()</code>的值来扩大元素的固有高度</li>
        <li>使用<code>morph(ease-function-name, start-value, end-value)</code> - 非线性缓冲函数</li>
        <li>使用<code>self.start-animation( 0.4s )</code>在连续的时间内同时改变动画的多个属性</li>
    </ul>
  <div .panel state="collapsed">
    <caption>分组一</caption>
    <div .details>CSSS!基于标准 CSS 语法，通常是以一个属性名称（该名称以惊叹号结束)标明触发的事件，
        以逗号分隔语句（可不是一般编程语言中使用的分号哦），而以分号结束语句块（不是大括号哦），
        这些非常规的规则为是了遵守 CSS 语法规范。
    </div>
  </div>
  <div .panel state="expanded">
    <caption>分组二</caption>
    <div .details>CSSS!基于标准 CSS 语法，通常是以一个属性名称（该名称以惊叹号结束)标明触发的事件，
        以逗号分隔语句（可不是一般编程语言中使用的分号哦），而以分号结束语句块（不是大括号哦），
        这些非常规的规则为是了遵守 CSS 语法规范。
    </div>
  </div>
</body>
</html>
```

横向扩展收缩动画

效果图


```
<html>
<head>
  <style>
    @const ANIMATION_STEP_MS: 8;
    div.playground
    {
      width:100%%;
      height:4em;
      overflow:hidden;
      background-color: white;
        style-set: "h-animate-on-caption-click"; // 见下文
      flow:horizontal;
```

```
    }
    div.playground > caption
    {
      background:orange;
      width:min-intrinsic;
      text-align:center;
    }
    div.playground > caption:hover
    {
      background:gold;
      transition:blend;
    }
    div.playground > div.expandable
    {
      border:1px solid black;
      overflow:hidden;
    }
    div.playground > div.expandable:animating,
    div.playground > div.expandable:expanded
    {
      background-color:floralwhite bisque bisque floralwhite;
    }
     @set h-animate-on-caption-click
{
          :root > caption
          {
      height:*;
            active-on!: panel = self.next(), // 下一个元素 (div)
                        panel:expanded ? (panel:collapsed = true) # (panel:expanded = true),
                        panel.start-animation();
          }
          :root > div /* 通常所有 div 内部折叠-不可见 */
          {
      height:*;
            visibility:collapse;
            overflow:hidden;
          }
    :root > div > *
    {
      width:max-intrinsic; // 防止换行
    }
    :root > div:animating,
          :root > div:expanded
          {
            width:100%%;
            visibility:visible;
          }
```

```
        :root > div:expanded
    {
      animation-start!: self::width = 1%% ;
      animation-end!:   self::width = 100%% ;
      animation-step!:  self::width < 100%% ?
              (
                  delta = limit( (100%% - self::width) * 0.15, 1%%, 8%% ),
                  self::width = self::width + delta,
                  self::width < 50%% ? self::opacity = float(self::width) / 50.0,
                  return @ANIMATION_STEP_MS
              );
    }
    :root > div:collapsed
        {
          animation-start!: self::width = 100%% ;
          animation-end!:   self::width = 0%%, self::opacity = 1.0;
          animation-step!:  self::width > 0%% ?
                  (
                      delta = limit( self::width * 0.15, 1%%, 8%% ),
                      self::width = self::width - delta,
                      self::width < 50%% ? self::opacity = float(self::width) / 50.0,
                      return @ANIMATION_STEP_MS
                  );
        }
    }
  </style>
<head>
<body>
  <div .playground>
    <caption>标题 1</caption>
    <div .expandable>
      面板 1
      <p>测试 测试 测试 测试 测试</p>
    </div>
    <caption>标题 2</caption>
    <div .expandable>
      面板 2
      <p>测试 测试 测试 测试 测试</p>
    </div>
    <caption>标题 3</caption>
    <div .expandable>
      面板 3
      <p>测试 测试 测试 测试 测试 测试</p>
    </div>
  </div>
  <div style="padding:20px">
    <h1>expand/collapse 动画简单示例</h1>
```

```
    <p>此示例依赖伸缩单位计算</p>
    <p>点击标题将显示动画</p>
  </div>
</body>
</html>
```

纵向扩展收缩动画

效果图


```
<html>
<head>
  <style>
    @const ANIMATION_STEP_MS: 8;
    div.playground
    {
      width:25%;
      height:*;
      overflow:hidden;
      background-color: white;
        style-set: "v-animate-on-caption-click"; // 见下文
    }
    div.playground > caption
    {
      background:orange;
    }
    div.playground > caption:hover
    {
      background:gold;
      transition:blend;
    }
    div.playground > div.expandable
    {
      border:1px solid black;
      overflow:hidden;
    }
    div.playground > div.expandable:animating,
    div.playground > div.expandable:expanded
    {
      background-color:floralwhite floralwhite bisque bisque;
```

```
    }
      @set v-animate-on-caption-click
    {
      :root > caption
      {
        active-on!: panel = self.next(), // 下一个元素(div)
        panel:expanded ? (panel:collapsed = true) # (panel:expanded = true),
        panel.start-animation();
      }
          :root > div /* 通常所有 div 内部折叠-不可见 */
          {
            width:100%%;
            visibility:collapse;
            overflow:hidden;
          }
    :root > div:animating,
          :root > div:expanded
          {
            height:100%%;
            visibility:visible;
          }
          :root > div:expanded
    {
      animation-start!: self::height = 1%% ;
      animation-end!:    self::height = 100%% ;
      animation-step!:   self::height < 100%% ?
        (
          delta = limit( (100%% - self::height) * 0.15, 1%%, 8%% ),
          self::height = self::height + delta,
          self::height < 50%% ? self::opacity = float(self::height) / 50.0,
          return @ANIMATION_STEP_MS
        );
    }
    :root > div:collapsed
          {
            animation-start!: self::height = 100%% ;
            animation-end!:    self::height = 0%%, self::opacity = 1.0;
            animation-step!:   self::height > 0%% ?
                    (
                        delta = limit( self::height * 0.15, 1%%, 8%% ),
                        self::height = self::height - delta,
                        self::height < 50%% ? self::opacity = float(self::height) / 50.0,
                        return @ANIMATION_STEP_MS
                    );
          }
     }
  </style>
```

```
<head>
<body style="flow:horizontal">
  <div .playground>
    <caption>标题 1</caption>
    <div .expandable>
      面板 1
      <p>测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试</p>
    </div>
    <caption>标题 2</caption>
    <div .expandable>
      面板 2
      <p>测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试</p>
    </div>
    <caption>标题 3</caption>
    <div .expandable>
      面板 3
      <p>测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试 测试</p>
    </div>
  </div>
  <div style="padding:20px">
    <h1>expand/collapse 动画简单示例</h1>
    <p>此示例依赖伸缩单位计算</p>
    <p>点击标题将显示动画</p>
  </div>
</body>
</html>
```

淡出动画

效果图

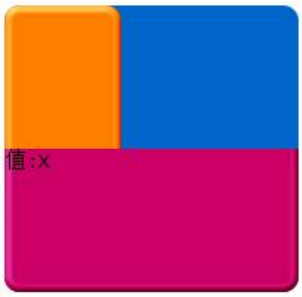
你好

```
<html>
  <head>
    <style>
    p#test[fade]
    {
      assigned!: self::opacity = 1.0, self.start-animation();
      animation-step!: self::opacity > 0
              ? ( self::opacity = self::opacity - 0.01, return 10)
              # ( self:value = self.fade, self.fade = "", return cancel );
    }
    p#test[fade=""]
```

```
    {
      color:red;
      assigned!: self::opacity = 0.01, self.start-animation();
      animation-step!: self::opacity < 1.0
              ? ( self::opacity = self::opacity + 0.01, return 10)
              # ( self.fade = "", return cancel );
    }
    button#do
    {
      transition:none;
      click! : $1( p#test ).fade="世界！";
    }
    </style>
  </head>
<body>
  <button #do>淡出 ！ </button>
  <p #test>你好</p>
</body>
</html>
```

仿进度条收缩动画

效果图

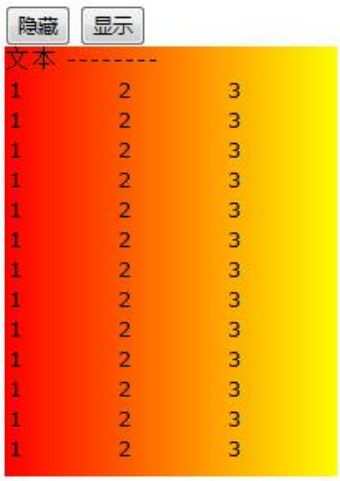



```
<html>
  <head>
    <style>
  #sandbox
  {
    background-image:url(images/slider-back.png);
    background-repeat:expand;
    background-position:10 10 10 10;
    width:50%;
    height:50%;
    margin:1*;
  }
  #sandbox>div
  {
    width:*;
    height:*;
    background-image:url(images/slider.png);
```

```
    background-repeat:expand;
    background-position:10 10 10 10;
    background-offset:0;
    assigned!: self.start-animation();
    animation-step!: self::background-offset-right > self.box-padding-width()? return cancel,
                        self::background-offset-right = self::background-offset-right + 2px, return 5;
  }
  #sandbox>div>div
  {
    width:*;
    margin-top:*;
    height:*;
    background-image:url(images/fore.png);
    background-repeat:expand;
    background-position:0 10 10 10;
  }
    </style>
  </head>
<body>
  <div #sandbox>
    <div><div>值:X</div></div>
  </div>
</body>
</html>
```

仿侧边栏平滑收缩/扩展动画

效果图




```
<html>
  <head>
    <style>
    /* 这些子类应用于水平可收缩面板。*/
    .expandable
    {
        height:                    100%%;
```

```
        width:                  200px;
        overflow:               hidden;
        background-color:       red yellow yellow red;
    }
    .expandable .content
    {
      width:                  200px;
      height:                 100%%;
    }
    .expandable:expanded
    {
        assigned!:              self::width = 0px, self.start-animation();
        animation-end!:         self::width = 200px;
        animation-step!:        self::width < 200px ? (self::width = self::width + 10px, return 8);
    }
    .expandable:collapsed
    {
        assigned!:              self::width = 200px, self.start-animation();
        animation-end!:         self::width = 0px;
        animation-step!:        self::width > 0px ? (self::width = self::width - 10px, return 8);
    }
    /* CSSS!脚本，隐藏面板 */
    .hidepanel
    {
        /* Event handler: */
        click!:         panel = $1(.expandable),
                        !panel:collapsed ? panel:collapsed = true;
    }
    /* CSSS!脚本，显示面板  */
    .showpanel
    {
        /* Event handler: */
        click!:         panel = $1(.expandable),
                        panel:collapsed? panel:expanded = true;
    }
    table
    {
      width:*;
      height:*;
      overflow:auto;
    }
    </style>
  </head>
<body>
  <button .hidepanel>隐藏</button>
  <button .showpanel>显示</button>
  <div .expandable>
```

```
    <div .content>
      文本 --------
      <table>
        <tr><td>1</td><td>2</td><td>3</td></tr>
        <tr><td>1</td><td>2</td><td>3</td></tr>
        <tr><td>1</td><td>2</td><td>3</td></tr>
        <tr><td>1</td><td>2</td><td>3</td></tr>
        <tr><td>1</td><td>2</td><td>3</td></tr>
        <tr><td>1</td><td>2</td><td>3</td></tr>
        <tr><td>1</td><td>2</td><td>3</td></tr>
        <tr><td>1</td><td>2</td><td>3</td></tr>
        <tr><td>1</td><td>2</td><td>3</td></tr>
        <tr><td>1</td><td>2</td><td>3</td></tr>
        <tr><td>1</td><td>2</td><td>3</td></tr>
        <tr><td>1</td><td>2</td><td>3</td></tr>
        <tr><td>1</td><td>2</td><td>3</td></tr>
      </table>
    </div>
  </div>
</body>
</html>
```

操作编辑框（到行头、行尾、弹出按钮）

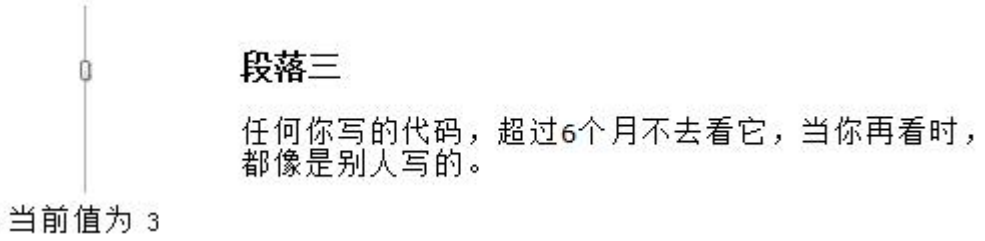
效果图


```
<html>
<head>
  <style>
    table.popup
    {
      display:none;
      background:white;
      border:1px solid red;
    }
    table.popup td
    {
      width:1.4em;
      border:1px solid black;
      text-align:center;
      cursor:pointer;
    }
    table.popup td:hover { background: yellow; }
```

```
    table.popup td:active { background: orange; }
  </style>
  <style #actions>
    table.popup td
    {
      behavior:clickable; // 绑定点击 behavior:clickable
      click!: $1(input#test).appendText( self:value );
    }
    table.popup td#backspace
    {
      click!: inp = $1(input#test),
              inp.setSelection( inp:value.length - 1,inp:value.length ),
              inp.insertText("");
    }
    button#to-first
    {
      click!: inp = $1(input#test),
              inp.setSelection(0,0),
              inp:focus = true;
    }
    button#to-last
    {
      click!: inp = $1(input#test),
              inp.setSelection(inp:value.length,inp:value.length),
              inp:focus = true;
    }
    input#test
    {
      active-off!: self.show-popup( $1(table.popup), 3 /*右下角描点*/, 9 /*右上角弹出位置描点*/ );
    }
  </style>
<head>
<body>
  <input #test type=text value="你好" /> <button #to-first>|&lt;</button><button #to-last>&gt;|</button>
  <p>点击编辑框查看弹出按钮</p>
  <table.popup>
    <tr><td>1</td><td>2</td><td>3</td></tr>
    <tr><td>4</td><td>5</td><td>6</td></tr>
    <tr><td>7</td><td>8</td><td>9</td></tr>
    <tr><td>0</td><td colspan=2 #backspace>&lt;-</td></tr>
  </table>
</body>
</html>
```

滑块条文本联动动画

效果图

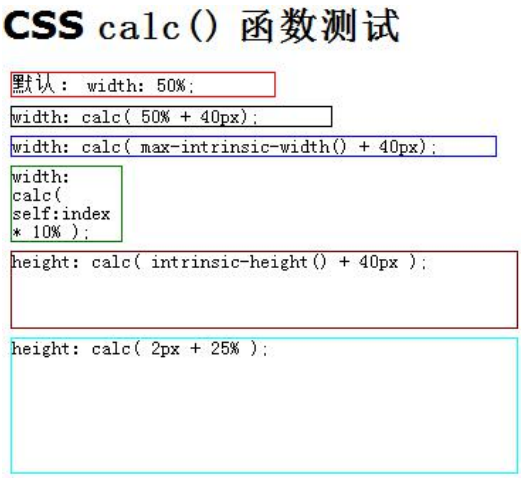



```
<html>
<head>
  <style>
    div.selector
    {
      flow:horizontal;
      height:300px;
    }
    widget#switch
    {
      height:100px;
      margin:*;
      assigned! :
        self.value-changed(); /* 当 value-changed 触发时调用函数*/
      value-changed! :
        $1(ul.bound>li[shown]).shown = null,
        $1(ul.bound>li:nth-child(<self:value>)).shown = true,
        $1(code#switch-value):value = self:value;
    }
    ul.bound > li { display:none }
    ul.bound > li[shown] { display:block; }
    caption { text-align:center; }
  </style>
<head>
<body>
  <div class="selector">
    <div style="width:100px">
      <widget id="switch" type="vslider" min=1 max=4 />
      <caption>当前值为 <code id="switch-value" /></caption>
    </div>
    <ul class="bound">
      <li shown="true">
        <h3>段落一</h3>
        电脑的确可以节约时间提高工效，比如说，玩翻纸牌游戏的时候你根本就用不着洗牌。
      </li>
      <li>
        <h3>段落二</h3>
        一个好的程序员是那种过单行线马路都要往两边看的人。
      </li>
      <li #test>
```

```
        <h3>段落三</h3>
        任何你写的代码，超过 6 个月不去看它，当你再看时，都像是别人写的。
      </li>
      <li>
        <h3>段落四</h3>
        一个人写的烂软件将会给另一个人带来一份全职工作。
      </li>
    </ul>
  </div>
</body>
</html>
```

CSS calc() 函数测试

效果图

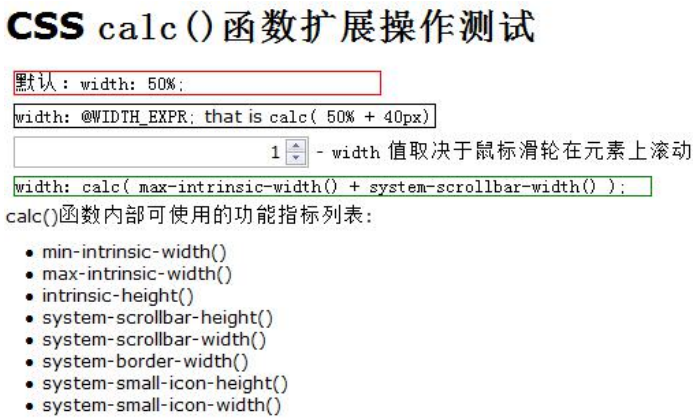



```
<html #test>
<head>
  <style>
    body { margin:0; padding:0; width:100%; height:100%; }
    div { margin:6px; }
    div#base
    {
      border:1px solid red;
      width: 50%;
    }
    div#test1
    {
      border:1px solid black;
      width: calc( 50% + 40px);
    }
    div#test2
    {
      border:1px solid blue;
      width: calc( max-intrinsic-width() + 40px );
    }
    div.test3
```

```
    {
      border:1px solid green;
      width: calc( self:index * 10% );
    }
    div#test4
    {
      border:1px solid maroon;
      height: calc( intrinsic-height() + 40px );
    }
    div#test5
    {
      border:1px solid cyan;
      height: calc( 2px + 25% );
    }
  </style>
</head>
<body #test>
  <H1>CSS <code>calc()</code> 函数测试</H1>
  <div #base>默认： <code>width: 50%;</code></div>
  <div #test1><code>width: calc( 50% + 40px);</code></div>
  <div #test2><code>width: calc( max-intrinsic-width() + 40px);</code></div>
  <div .test3><code>width: calc( self:index * 10% );</code></div>
  <div #test4><code>height: calc( intrinsic-height() + 40px );</code></div>
  <div #test5><code>height: calc( 2px + 25% );</code></div>
</body>
</html>
```

CSS calc() 函数扩展操作测试

效果图

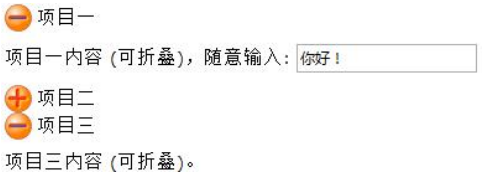



```
<html #test>
<head>
  <style>
    @const WIDTH_EXPR: calc( 50% + 40px);
    body { margin:0; padding:0; width:100%; height:100%; }
    div { margin:6px; }
    div#base
```

```
    {
      border:1px solid red;
      width: 50%;
    }
    div#test1
    {
      border:1px solid black;
      width: @WIDTH_EXPR;
    }
    input#test2
    {
      width: calc( 200px + self:value * 10px );
      value-changed!: self.update(); /* force width:calc() re-evalutation */
    }
    div#test3
    {
      border:1px solid green;
      width: calc( max-intrinsic-width() +  system-scrollbar-width() );
    }
  </style>
</head>
<body #test>
  <H1>CSS <code>calc()</code>函数扩展操作测试</H1>
  <div #base>默认：<code>width: 50%;</code></div>
  <div #test1><code>width: @WIDTH_EXPR;</code> that is <code>calc( 50% + 40px)</code></div>
  <div><input type="number" #test2 value=1 min=1 max=10 step=1 /> - <code>width</code> 值取决于鼠标滑轮在元素上滚动</div>
  <div #test3><code>width: calc( max-intrinsic-width() +  system-scrollbar-width() );</code></div>
  calc()函数内部可使用的功能指标列表:
  <ul>
    <li>min-intrinsic-width()</li>
    <li>max-intrinsic-width()</li>
    <li>intrinsic-height()</li>
    <li>system-scrollbar-height()</li>
    <li>system-scrollbar-width()</li>
    <li>system-border-width()</li>
    <li>system-small-icon-height()</li>
    <li>system-small-icon-width()</li>
  </ul>
</body>
</html>
```

可折叠列表一

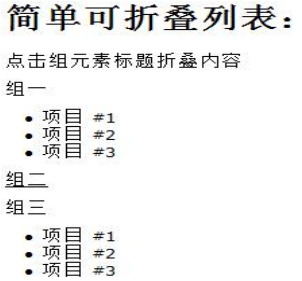
效果图




```
<html>
  <head>
    <style>
  img.icon
  {
    active-on!: $1p(li):collapsed = ! $1p(li):collapsed;
    vertical-align:middle;
    margin-right: 4px;
    foreground-image:url(images/minus.png);
    cursor:pointer;
  }
  li { display:block; }
  li > p { display:block }
  li:collapsed > p { display:none; }
  li:collapsed img.icon { foreground-image:url(images/plus.png); }
    </style>
  </head>
<body>
  <ul>
    <li><img class="icon"/>项目一
      <p>项目一内容 (可折叠)，随意输入: <input type="text" value="你好！" /></p>
    </li>
    <li><img class="icon"/>项目二
      <p>项目二内容 (可折叠)。</p>
    </li>
    <li><img class="icon"/>项目三
      <p>项目三内容 (可折叠)。</p>
    </li>
  </ul>
</body>
</html>
```

可折叠列表二

效果图



```
<html>
<head>
  <style>
    p { active-on!: self:checked = !self:checked; }
    p:hover { text-decoration:underline; cursor:pointer; }
    p + ul { display:block; }
    p:checked + ul { display:none; }
  </style>
<head>
<body>
  <h1>简单可折叠列表：</h1>
  点击组元素标题折叠内容
  <p>组一</p>
  <ul>
    <li>项目 #1</li>
    <li>项目 #2</li>
    <li>项目 #3</li>
  </ul>
  <p>组二</p>
  <ul>
    <li>项目 #1</li>
    <li>项目 #2</li>
    <li>项目 #3</li>
  </ul>
  <p>组三</p>
  <ul>
    <li>项目 #1</li>
    <li>项目 #2</li>
    <li>项目 #3</li>
  </ul>
</body>
</html>
```

可折叠列表三

效果图同 可折叠列表二

```
<html>
<head>
  <style>
    p { behavior:check; }
    p:hover { text-decoration:underline; }
    p + ul { display:block; }
    p:checked + ul { display:none; }
  </style>
<head>
<body>
```

```
  <h1>简单可折叠列表：</h1>
  点击组元素标题折叠内容
  <p>组一</p>
  <ul>
    <li>项目 #1</li>
    <li>项目 #2</li>
    <li>项目 #3</li>
  </ul>
  <p>组二</p>
  <ul>
    <li>项目 #1</li>
    <li>项目 #2</li>
    <li>项目 #3</li>
  </ul>
  <p>组三</p>
  <ul>
    <li>项目 #1</li>
    <li>项目 #2</li>
    <li>项目 #3</li>
  </ul>
</body>
</html>
```

仿选择下拉框

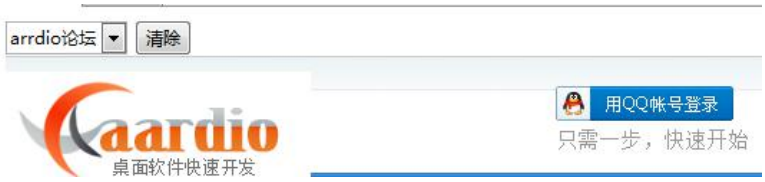
效果图


```
<html>
<head>
  <style>
    widget
    {
      width:100px;
      border:1px solid;
      padding:2px;
      position:relative;
      background-image: url(stock:arrow-down);
      background-repeat: no-repeat;
      background-position-top: 50%;
      background-position-right: 3px;
      active-on!: self.popup-shown? (self.popup-shown = null) # (self.popup-shown = true, self.$1(popup):focus = true);
      focus-on!: ; // 定义 focus-on 是为了表明它是可获得焦点的
    }
    widget > popup
```

```
    {
      behavior:select;
      display:none;
      position:absolute;
      border:1px solid black;
      left:-1px;
      right:-1px;
      height:min-intrinsic;
    }
    widget[popup-shown] > popup
    {
      display:block;
      outline:3px glow silver 1px;
      outline-shift:3px;
    }
    widget > popup > option:current
    {
      color:highlighttext;
      background:highlight;
    }
    widget > popup
    {
      active-off!: self.parent().child(1):value = self.$1(option:current):value, // 将当前选择的文本复制到标题
                   self.parent():focus = true, // 并将焦点设置在父元素上
                   self.parent().popup-shown = null; // 同时隐藏弹出窗口
      focus-off!:  self.parent().popup-shown = null; // 隐藏弹出窗口
      key-off! :  key-code() == 'RETURN'?  self.active-off();
    }
  </style>
<head>
<body>
  <widget>
    请选择...
    <popup>
      <option>标题一</option>
      <option>标题二</option>
      <option>标题三</option>
    </popup>
  </widget>
</body>
</html>
```

框架加载指定网页

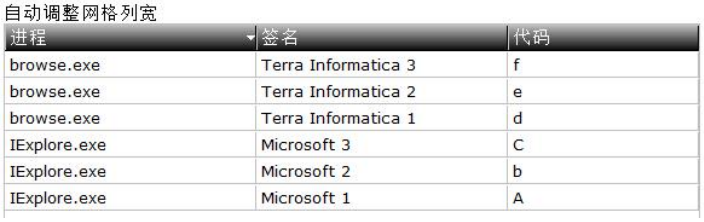
效果图



```
<html>
<head>
  <style>
    frame#view
    {
      height:*;
      width:*;
    }
    frame#view:busy
    {
      foreground-image: url(res:activity.gif);
      foreground-repeat:no-repeat;
      foreground-position:top left;
      foreground-offset: 20px;
    }
    select#url-to-load
    {
      value-changed!: $1(frame#view).load( self:value );
    }
    button#clear
    {
      click!: $1(frame#view).clear();
    }
  </style>
</head>
<body>
  <select #url-to-load novalue="选择站点">
    <option value="http://bbs.aardio.com/forum.php">arrdio 论坛</option>
    <option value="https://msdn.microsoft.com/library">msdn 文档</option>
    <option value="http://bbs.csdn.net/home">csdn 论坛</option>
  </select> <button #clear>清除</button>
  <frame #view />
</body>
</html>
```

表格操作（简单排序、自适应列宽）

效果图



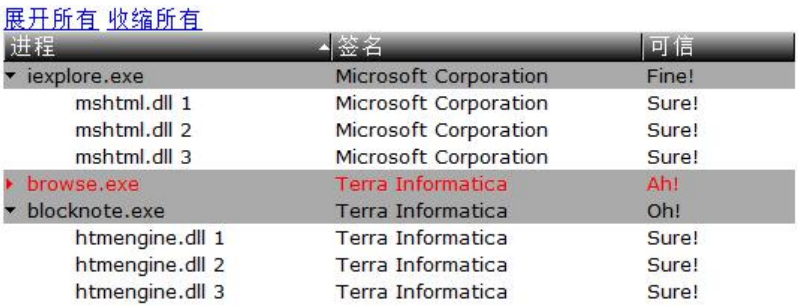

```
<html>
  <head>
    <style>
```

```
table
{
  behavior:grid column-resizer;
  overflow:auto;
  width:*; height:*;
  border-spacing:0;
  background-color:white;
  border:1px solid silver;
}
table td { white-space:nowrap; padding:2px 3px; border:1px solid; border-color: transparent silver silver transparent;}
table tr.header { background-color:silver silver black black; color:white;  }
table tr.header td { border-color: silver black black silver; foreground-position:100% 50%; foreground-repeat:no-repeat;}
table tr.header td:hover { border-bottom-color: orange; }
table td:nth-child(1) { min-width:100px; width:200px;}
table td:nth-child(2) { min-width:50px; width:200px; }
table td:nth-child(3) { min-width:50px; width:*; }
tr:current { color:red; }
/* 行为样式 */
/* 在列标题上单击 */
tr.header > td
{
  active-on! :
    self.parent().$1( td[sorted] ).sorted = null,    // 删除前面行的 td@sorted 标签
    self.sorted = "asc",                             // 设置元素@sorted="asc"
    ncolumn = self:index,                            // 在比较器中存储列供以后使用
    is_less = @(row1, row2)                          // 比较函数本身
      row1.child(ncolumn):value.toLower() < row2.child(ncolumn):value.toLower() ? true #
      row1.child(ncolumn):value.toLower() > row2.child(ncolumn):value.toLower() ? false #
      row1:index < row2:index,
    self.$1p(table).$(tr:not(.header)).sort(is_less);        // 排序
}
tr.header > td[sorted="asc"]
{
  foreground-image: url(stock:arrow-down);
  active-on! :
    self.sorted = "desc",
    ncolumn = self:index,
    is_less = @(row1, row2)
      row2.child(ncolumn):value.toLower() < row1.child(ncolumn):value.toLower() ? true #
      row2.child(ncolumn):value.toLower() > row1.child(ncolumn):value.toLower() ? false #
      row1:index < row2:index,
    self.$1p(table).$(tr:not(.header)).sort(is_less);
}
tr.header > td[sorted="desc"]
{
```

```
        foreground-image: url(stock:arrow-up);
      }
    </style>
  </head>
<body>
  自动调整网格列宽
  <table fixedrows=1 fixedlayout >
    <tr .header ><td>进程</td><td>签名</td><td>代码</td></tr>
    <tr ><td>browse.exe</td><td>Terra Informatica 3</td><td>f</td></tr>
    <tr ><td>browse.exe</td><td>Terra Informatica 2</td><td>e</td></tr>
    <tr ><td>browse.exe</td><td>Terra Informatica 1</td><td>d</td></tr>
    <tr ><td>IExplore.exe</td><td>Microsoft 3</td><td>C</td></tr>
    <tr ><td>IExplore.exe</td><td>Microsoft 2</td><td>b</td></tr>
    <tr ><td>IExplore.exe</td><td>Microsoft 1</td><td>A</td></tr>
  </table>
</body>
</html>
```

扩展表格列表

效果图




```
<html>
  <head>
    <style>
      table
      {
        behavior:grid column-resizer;
        overflow:auto;
        width:*; height:*;
        border-spacing:0;
        background-color:white;
      }
      table td { white-space:nowrap; padding:1px 3px; }
      table tr.header { background-color:silver silver black black; color:white; }
      table tr.header td { border:1px solid; border-color: silver black black silver; }
      table td:nth-child(1) { min-width:100px; width:200px;}
      table td:nth-child(2) { min-width:50px; width:200px; }
      table td:nth-child(3) { min-width:50px; width:*; }
      tr:current { color:red; }
```

```
    tr.exe { background-color:#AAA; cursor:pointer; }
    tr.exe > td:nth-child(1) { padding-left:16px; foreground-image:url(stock:arrow-down); foreground-position:1p
x 50%; foreground-repeat:no-repeat; }
    tr.exe[collapsed] > td:nth-child(1) { foreground-image:url(stock:arrow-right); }
    tr.dll > td:nth-child(1) { padding-left:48px; }
    tr.dll[collapsed] { display:none; }
    /* 行为样式 */
    tr.exe
    {
      double-click! : self.collapsed = self.collapsed? null # true, // 触发器的折叠属性
                      $( tr.dll[process="< self.process >"] ).collapsed = self.collapsed;
                                  // 展开/折叠所有 tr.dll 具有相同进程名称的行
    }
    tr.exe > td:nth-child(1)
    {
      active-on!: is-on-icon() ? (self.parent().double-click(), return cancel);
    }
    table
    {
      assigned! :
        self:focus = true,
        $1( tr.exe ):current = true; // 默认情况下第一行 :current 属性
      key-on! : !key-code('LEFT','RIGHT')? return, // 如果不是 VK_LEFT 或 VK_RIGHT 键直接返回
                cur = self.$1(tr.dll:current),
                cur = cur? self.$1(tr.exe[process="< cur.process >"]) #
                        self.$1(tr.exe:current),
                !cur? return,  // 没有返回, 没有 current 项目
                // cur 当前包含了 tr.exe，我们需要他们折叠/收缩
                key-code() == 'LEFT'?  // VK_LEFT?
                  (
                    self.$1(tr:current):current = false, // 移除匹配到的第一个 current
                    cur:current = true,                  //把它当作一个新的 current 设置
                    self.$(tr[process="< cur.process >"]).collapsed = true // 收缩所有组
                  ),
                key-code() == 'RIGHT'? // VK_RIGHT?
                    self.$(tr[process="< cur.process >"]).collapsed = null, // 展开所有组
                return cancel; // 返回 'cancel' 值 - 不需要进一步的处理，因为已经处理过 key 了。
    }
    a#expand-all
    {
      click! : $( tr.dll, tr.exe ).collapsed = null;
    }
    a#collapse-all
    {
      click! :  $( tr.dll, tr.exe ).collapsed = true, // 全部添加 [collapsed="true"] 属性
                cur_dll = $1( tr.dll:current ), // 完整性检查，如果我们隐藏当前行...
                cur_dll?
```

```
                (
                  cur_dll:current = false,
                  $1( tr.exe[process="< cur_dll.process >"] ):current = true // 把当前的 tr.exe 作为一个 current
                );
    }
    /* 在列标题上单击 */
    tr.header > td:nth-child(1)
    {
      foreground-position:100% 50%; foreground-repeat:no-repeat;
      active-on! :
        self.parent().$1( td[sorted] ).sorted = null,    // 删除前面行的 td@sorted 标签
        self.sorted = "asc",                             // 设置元素@sorted="asc"
        ncolumn = self:index,                            // 在比较器中存储列供以后使用
        is_less = @(row1, row2)                          // 比较函数本身
          row1.process < row2.process ? true #
          row1.process > row2.process ? false #
          row1:index < row2:index,
        $(table>tr:not(.header)).sort(is_less);          // 排版
    }
    tr.header > td[sorted="asc"]:nth-child(1)
    {
      foreground-image: url(stock:arrow-down);
      active-on! :
        self.sorted = "desc",
        ncolumn = self:index,
        is_less = @(row1, row2)
          row2.process < row1.process ? true #
          row2.process > row1.process ? false #
          row1:index < row2:index,
        $(table>tr:not(.header)).sort(is_less);
    }
    tr.header > td[sorted="desc"]:nth-child(1)
    {
      foreground-image: url(stock:arrow-up);
    }
  </style>
 </head>
<body>
 <a #expand-all href=#>展开所有</a> <a #collapse-all href=#>收缩所有</a>
 <table fixedrows=1 fixedlayout #test>
  <tr .header ><td>进程</td><td>签名</td><td>可信</td></tr>
  <tr .exe process="browse.exe"><td>browse.exe</td><td>Terra Informatica</td><td>Ah!</td></tr>
    <tr .dll process="browse.exe"><td>htmlayout.dll 1</td><td>Terra Informatica</td><td>Sure!</td></tr>
    <tr .dll process="browse.exe"><td>htmlayout.dll 2</td><td>Terra Informatica</td><td>Sure!</td></tr>
    <tr .dll process="browse.exe"><td>htmlayout.dll 3</td><td>Terra Informatica</td><td>Sure!</td></tr>
  <tr .exe process="iexplore.exe"><td>iexplore.exe</td><td>Microsoft Corporation</td><td>Fine!</td></tr>
    <tr .dll process="iexplore.exe"><td>mshtml.dll 1</td><td>Microsoft Corporation</td><td>Sure!</td></t
```

```
r>
      <tr .dll process="iexplore.exe"><td>mshtml.dll 2</td><td>Microsoft Corporation</td><td>Sure!</td></t
r>
      <tr .dll process="iexplore.exe"><td>mshtml.dll 3</td><td>Microsoft Corporation</td><td>Sure!</td></t
r>
    <tr .exe process="blocknote.exe"><td>blocknote.exe</td><td>Terra Informatica</td><td>Oh!</td></tr>
      <tr .dll process="blocknote.exe"><td>htmengine.dll 1</td><td>Terra Informatica</td><td>Sure!</td></t
r>
      <tr .dll process="blocknote.exe"><td>htmengine.dll 2</td><td>Terra Informatica</td><td>Sure!</td></t
r>
      <tr .dll process="blocknote.exe"><td>htmengine.dll 3</td><td>Terra Informatica</td><td>Sure!</td></t
r>
  </table>
</body>
</html>
```

高亮链接

效果图

盖茨夫人接受采访时说：我们家从来不用苹果的产品，甚至连苹果都不吃。坐一旁的乔布斯不屑一顾地说到：切~那有什么了不起，我们家连窗户都没有... 扎克伯格听了，说：那你们敢不要脸吗？

鼠标在此悬停 高亮上面的链接

```
<html>
<head>
  <style>
    a#master
    {
      border:1px solid;
      hover-on!: $(p.slave a):hover = true;
      hover-off!: $(p.slave a):hover = false;
    }
  </style>
<head>
<body>
  <p class="slave">
<a href="#">盖茨夫人</a>接受采访时说：我们家从来不用苹果的产品，甚至连苹果都不吃。
坐一旁的<a href="#">乔布斯</a>不屑一顾地说到：切~那有什么了不起，我们家连窗户都没有...
<a href="#">扎克伯格</a>听了，说：那你们敢不要脸吗?
  </p>
  <a id="master">鼠标在此悬停</a> 高亮上面的链接
</body>
</html>
```

CSSS! 键盘演示

效果图


```
<html>
<head>
  <style>
  table
      {
        assigned! :
          $1( td ):current = true; // 第一个 TD，默认:current
        focus-on! :
          $1( td:current )? self.assigned();
        key-on! : !key-code('LEFT','RIGHT','UP','DOWN' )? return, // 如果不是 VK_LEFT 或 VK_RIGHT 键直接返回
                  cur = self.$1(td:current), !cur? return,  // 无返回，无 current 项目
                  // cur 当前包含 td:current
                  ridx = cur.parent():index,
                  cidx = cur:index,
                  key-code() == 'LEFT'?  ( cidx = cidx - 1, cidx < 1? cidx = cur.parent().children() ),
                  key-code() == 'RIGHT'? ( cidx = cidx + 1, cidx > cur.parent().children()? cidx = 1 ),
                  key-code() == 'UP'?    ( ridx = ridx - 1, ridx < 1? ridx = cur.parent().parent().children() ),
                  key-code() == 'DOWN'?  ( ridx = ridx + 1, ridx > cur.parent().parent().children()? ridx = 1 ),
                  cur:current = false,   //不再是 current
                  self.$1(tr:nth-child(< ridx >) > td:nth-child(< cidx >) ):current = true,
                  return cancel; // 返回 'cancel' 值 - 不需要进一步的处理，因为已经处理过 key 了。
      }
  table td
  {
    border:1px solid blue;
    padding:20px;
    active-on!: $1(td:current):current = false, self:current = true;
  }
  table td:current { background-color:yellow; }
  table:focus td:current { background-color:orange; color:red; }
  </style>
<head>
<body>
  <h1>CSSS! 键盘演示</h1>
  单击表或按方向键。
```

```
  <table>
    <tr><td>1</td><td>2</td><td>3</td></tr>
    <tr><td>4</td><td>5</td><td>6</td></tr>
    <tr><td>7</td><td>7</td><td>9</td></tr>
    <tr><td>10</td><td>11</td><td>12</td></tr>
  </table>
</body>
</html>
```

鼠标悬停标签高亮编辑框

效果图


```
<html>
  <head>
    <style>
    [for]
    {
      hover-on!  : $1(#< self.for >).highlight = true;
      hover-off! : $1(#< self.for >).highlight = null;
      active-on! : $1(#< self.for >):focus = true;
      cursor:pointer;
    }
    input[highlight]
    {
      outline: 4px glow blue 1px;
    }
    </style>
  </head>
<body>
  <table border="1" width="100%" #test>
    <tr>
      <td><label for="first">标签 #1</label></td>
      <td><input type="text" id="first" /></td></tr>
    <tr>
      <td for="second">单元格作为标签</td>
      <td><input type="text" id="second" /></td></tr>
    <tr for="third">
      <td>整行作为一个标签</td>
      <td><input type="text" id="third" /></td></tr>
  </table>
  <label for="fourth">标签 #4</label><input type="text" id="fourth" />
</body>
```

```
</html>
```

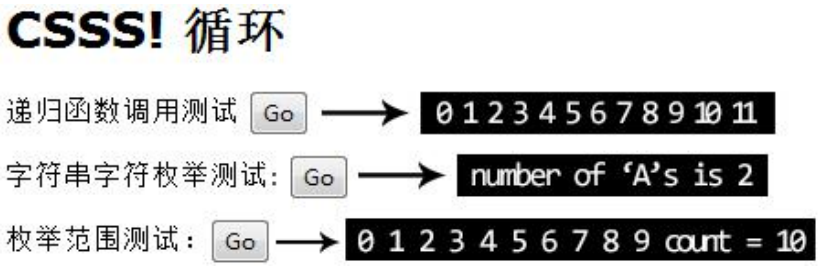
使用 DOM 中定义的字符串

效果图


```
<html>
  <head>
    <meta id="INITIAL_TEXT" text="世界" />
    <style>
      span.greeting
      {
        font-weight:bold;
        assigned!: self:value = $1(#INITIAL_TEXT).text;
      }
    </style>
  </head>
<body>
  这个示例演示了使用 DOM 中定义的字符串值。
  <p><span .greeting />，你好！ </p>
</body>
</html>
```

CSSS!循环

效果图




```
<html>
<head>
  <style>
    button#test1
    {
        click!:
         Foo = null, //递归调用前声明是必须
         Foo = @(n)
              (
                debug(n),
                (n > 10)? return,
                Foo(n + 1)
              ),
```

```
        Foo(0);
    }
    button#test2
    {
        click!:
         count = 0,
         "ABCDEFABC" -> @(c)(c == 'A'? count = count + 1),
         debug("number of 'A's is ", count );
    }
    button#test3
    {
        click!:
         count = 0,
         0..9 -> @(i) (count = count + 1, debug(i)), // 遍历范围,输出将包含数字 0...9
         debug("count=",count);
    }
  </style>
</head>
<body>
  <h1>CSSS! 循环</h1>
  <p>递归函数调用测试 <button #test1>Go</button></p>
  <p>字符串字符枚举测试: <button #test2>Go</button></p>
  <p>枚举范围测试: <button #test3>Go</button></p>
</body>
</html>
```

进度条/自定义进度条

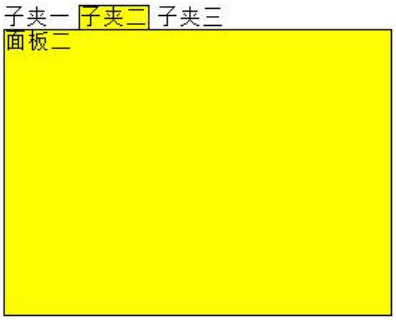
效果图


```
<HTML>
    <HEAD>
        <STYLE>
      input#val
      {
        value-changed!: $1(input#pro).requested-value = self:value;
      }
      input#pro[requested-value]
      {
        assigned!: self.start-animation();
```

```
        animation-step!:
            rqval = int(self.requested-value),
            rqval == int(self:value) ? return,
            self:value = self:value + (rqval < self:value? -1 # 1),
            //debug(self:value),
            return 20;
        animation-end!: self.requested-value = null;
    }
    input#val2
    {
        value-changed!: $1(span#pro2)::foreground-offset-right = 100 - self:value;
    }
    span#pro2
    {
        vertical-align:middle;
        foreground: url(images/anim-bar.gif) no-repeat;
        display:inline-block;
        foreground-offset-right:90;
        width:100px;
        height:6px;
        border:1px solid;
    }
     </STYLE>
  </HEAD>
  <BODY>
    <h1>进度条 1</h1>
    改变以下值：<input #val type="number" value=10 maxvalue=100 minvalue=0 step="10" /> 观察动画改变:
    <input #pro type="progress" maxvalue="100" value="0"/>
    <h1>进度条 2, 动画</h1>
    改变以下值：<input #val2 type="number" value=10 maxvalue=100 minvalue=0 step="10" />观察动画改变:
    <span #pro2 />
  </BODY>
  </BASEFONT>
</HTML>
```

选择夹一

效果图



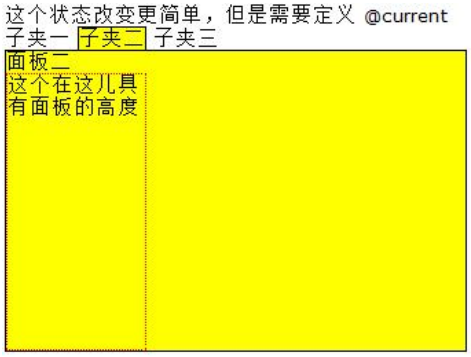

```
<html>
  <head>
```

```
    <style>
/*
  div.tab-strip span[current]
  {
    assigned!:  self:current = true,
                $1(div.panels>div:nth-child(< self:index >)):expanded = true );
  }
  代码另一种改变状态的实现方法 (self.current?):
*/
  div.tab-strip span
  {
    assigned! : self.current ? /* 如果有'current'属性定义则下一块区域作为一个整体: */
                ( self:current = true,
                  $1(div.panels>div:nth-child(< self:index >)):expanded = true );

    active-on! :  $1(div.tab-strip span:current):current = false,
                  $1(div.panels>div:expanded):expanded = false,
                  self:current = true,
                  $1(div.panels>div:nth-child(< self:index >)):expanded = true;
  }
  div.tab-strip span:current { background: yellow; border:1px solid; border-bottom:none; }
  div.panels > div { display: none; }
  div.panels > div:expanded { display: block; background: yellow; border:1px solid; height:200px; }
    </style>
  </head>
<body>
  <div class="tab-strip">
    <span #i1 >子夹一</span>
    <span #i2 current>子夹二</span>
    <span #i3 >子夹三</span>
  </div>
  <div class="panels">
    <div>
      面板一
    </div>
    <div>
      面板二
    </div>
    <div>
      面板三
    </div>
  </div>
</body>
</html>
```

选择夹二

效果图



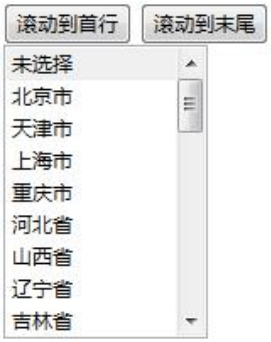

```
<html>
  <head>
    <style>

  div.tab-strip span
  {
    active-on!: /* 移除 @current 属性 */
                $1(div.tab-strip span[current]).current = null,
                /* 重新设定 @current 属性 */
                self.current = true;
  }
  div.tab-strip span[current]
  {
    assigned!:  /* remove @current from card panel */
                $1(div.panels > div[current]).current = null,
                /* set @current to card panel by this index */
                $1(div.panels > div:nth-child(< self:index >)).current = true;
  }
  div.tab-strip span[current] { background: yellow; border:1px solid; border-bottom:none; }
  div.panels > div { visibility: hidden; }
  div.panels > div[current] { visibility: visible; height:*; }
  div.panels
  {
    background: yellow;
    border: 1px solid;
    flow: stack;
  }
    </style>
  </head>
<body>
  这个状态改变更简单，但是需要定义 @current
  <div class="tab-strip">
    <span current>子夹一</span>
    <span>子夹二</span>
    <span>子夹三</span>
  </div>
  <div class="panels">
    <div>
```

```
      面板一
    </div>
    <div>
      面板二
      <div style="width:100px; height:200px; border:1px dotted red">
        这个在这儿具有面板的高度
      </div>
    </div>
    <div>
      面板三
    </div>
  </div>
</body>
</html>
```

列表滚动

效果图

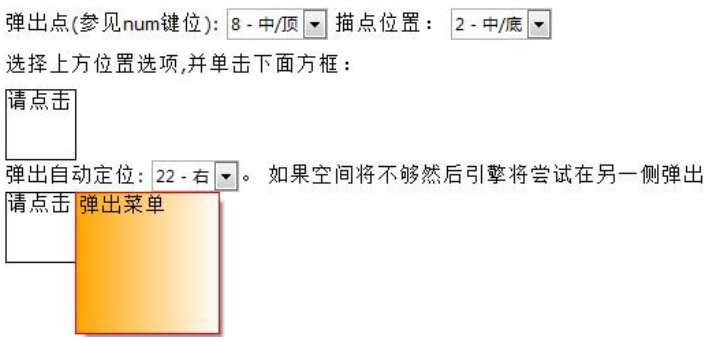



```
<html>
<head>
  <style>
  widget#select
  {
    overflow-y:auto;
    height:*;
  }
  </style>
<head>
<body>
  <button style="click!:$1(#select>:first-child).scroll-to-view()">滚动到首行</button>
  <button style="click!:$1(#select>:last-child).scroll-to-view()">滚动到末尾</button>
  <widget type="select" #select size=10>
    <OPTION VALUE=0 SELECTED>未选择</OPTION>
    <OPTION VALUE=1 >北京市</OPTION>
    <OPTION VALUE=2 >天津市</OPTION>
    <OPTION VALUE=3 >上海市</OPTION>
    <OPTION VALUE=4 >重庆市</OPTION>
    <OPTION VALUE=5 >河北省</OPTION>
    <OPTION VALUE=6 >山西省</OPTION>
```

```
    <OPTION VALUE=7 >辽宁省</OPTION>
    <OPTION VALUE=8 >吉林省</OPTION>
    <OPTION VALUE=9 >黑龙江省</OPTION>
    <OPTION VALUE=10 >江苏省</OPTION>
    <OPTION VALUE=11 >浙江省</OPTION>
    <OPTION VALUE=12 >安徽省</OPTION>
    <OPTION VALUE=13 >福建省</OPTION>
    <OPTION VALUE=14 >江西省</OPTION>
    <OPTION VALUE=15 >山东省</OPTION>
    <OPTION VALUE=16 >河南省</OPTION>
    <OPTION VALUE=17 >湖北省</OPTION>
    <OPTION VALUE=18 >湖南省</OPTION>
    <OPTION VALUE=19 >广东省</OPTION>
    <OPTION VALUE=20 >海南省</OPTION>
    <OPTION VALUE=21 >四川省</OPTION>
    <OPTION VALUE=22 >贵州省</OPTION>
    <OPTION VALUE=23 >云南省</OPTION>
    <OPTION VALUE=24 >陕西省</OPTION>
    <OPTION VALUE=25 >甘肃省</OPTION>
    <OPTION VALUE=26 >青海省</OPTION>
    <OPTION VALUE=27 >台湾省</OPTION>
    <OPTION VALUE=28 >内蒙古自治区</OPTION>
    <OPTION VALUE=29 >广西壮族自治区</OPTION>
    <OPTION VALUE=30 >西藏自治区</OPTION>
    <OPTION VALUE=31 >宁夏回族自治区</OPTION>
    <OPTION VALUE=32 >新疆维吾尔自治区</OPTION>
    <OPTION VALUE=33 >香港特别行政区</OPTION>
    <OPTION VALUE=34 >澳门特别行政区</OPTION>
  </widget>
</body>
</html>
```

显示弹出框

效果图




```
<html>
<head>
  <style>
    popup#our,
```

```
  popup#our2
  {
    background-color: white;
    border:1px solid red;
    width:100px;
    height:100px;
    cursor:pointer;
  }
  div#anchor,
  div#anchor2
  {
    border:1px solid black;
    width:50px;
    height:50px;
    cursor:pointer;
  }
</style>
<style #actions>
  div#anchor
  {
    behavior: button;
    click!: anchor-point = $1(select#anchor-point):value,
            popup-point = $1(select#popup-point):value,
            self.show-popup( $1(popup#our), anchor-point, popup-point );
  }
  div#anchor2
  {
    behavior: button;
    click!: popup-pos = $1(select#popup-auto):value,
            self.show-popup( $1(popup#our2), popup-pos );
  }
  popup#our,
  popup#our2
  {
     active-off!: self:popup = false;
  }
  popup#our2
  {
    background-color: orange white white orange;
  }
  popup#our2:popup
  {
    assigned!: self.on-the-right = self.x-parent() > 0 ? true # null;
               //self.on-the-right = $1(div#anchor2).x-view() < self.x-view() ? true # null; - 如果不是父窗口...
  }
  popup#our2[on-the-right]
  {
```

```
        background-color: white orange orange white;
    }
  </style>
<head>
<body>

  弹出点(参见 num 键位):
  <select #popup-point>
    <option value=1>1 - 左/底</option>
    <option value=2>2 - 中/底</option>
    <option value=3>3 - 右/底</option>
    <option value=4>4 - 左/中</option>
    <option value=5>5 - 中/中</option>
    <option value=6>6 - 右/中</option>
    <option value=7>7 - 左/顶</option>
    <option value=8 selected>8 - 中/顶</option>
    <option value=9>9 - 右/顶</option>
  </select> 描点位置:
  <select #anchor-point>
    <option value=1>1 - 左/底</option>
    <option value=2 selected>2 - 中/底</option>
    <option value=3>3 - 右/底</option>
    <option value=4>4 - 左/中</option>
    <option value=5>5 - 中/中</option>
    <option value=6>6 - 右/中</option>
    <option value=7>7 - 左/顶</option>
    <option value=8>8 - 中/顶</option>
    <option value=9>9 - 右/顶</option>
  </select>
  <p>选择上方位置选项,并单击下面方框: </p>
   <div#anchor>
     请点击
   </div>
   <popup#our>
     弹出菜单
   </popup>
  弹出自动定位:
  <select #popup-auto>
    <option value=20>20 - 左</option>
    <option value=22>22 - 右</option>
    <option value=8>8 - 顶</option>
    <option value=2>2 - 底</option>
  </select>。 如果空间将不够然后引擎将尝试在另一侧弹出
   <div#anchor2>
     请点击
   </div>
   <popup#our2>
```

```
        弹出菜单
    </popup>
</body>
</html>
```

取自身窗口大小

效果图


```
<html>
<head>
  <style>
    body
    {
      size-changed!: $1(span#out):value = self.box-border-width() + "," + self.box-border-height();
    }
  </style>
<head>
<body>
  Body 尺寸大小:<span #out></span> 像素。
</body>
</html>
```

纵向滑动条

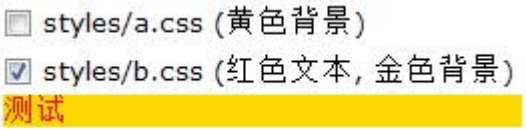
效果图


```
<html>
<head>
  <style>
    widget#bottles
    {
      height:200px;
      margin:*;
      assigned! :
          self.value-changed(); /* 当触发 value-changed 事件句柄时调用函数 */
      value-changed! :
          $1(code#bottles-value):value =
            self:value == 0 ? "没有瓶子" #
```

```
            self:value == 1 ? "有 1 个瓶子" #
                              "有 " + self:value + " 瓶子";
    }
    caption { text-align:center; }
  </style>
<head>
<body>
    <div style="width:200px">
      <widget id="bottles" type="vslider" min=0 max=99 value=0 />
      <caption>这里 <code id="bottles-value" /> 在墙上</caption>
    </div>
</body>
</html>
```

CSS 样式修改

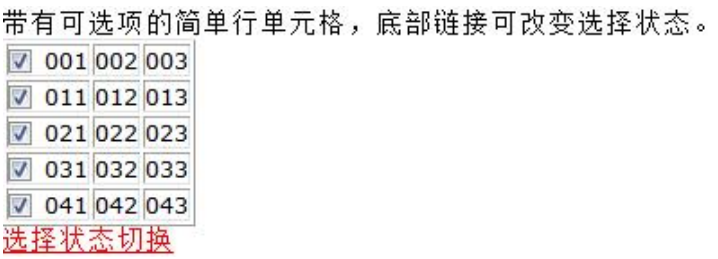
效果图




```
<html>
<head>
  <style src="styles/a.css" #style-set-a disabled />
  <style src="styles/b.css" #style-set-b />
  <style>
    button#activate-a
    {
      click!: style-set = $1(#style-set-a),
      style-set:disabled = !self:value;
    }
    button#activate-b
    {
      click!: style-set = $1(#style-set-b),
      style-set:disabled = !self:value;
    }
  </style>
</head>
<body>
  <button #activate-a type="checkbox">styles/a.css (黄色背景)</button>
  <button #activate-b type="checkbox" checked>styles/b.css (红色文本, 金色背景)</button>
  <div #test>
    测试
  </div>
</body>
</html>
```

表格全选/反选

效果图




```
<html>
  <head>
    <style>
      table#checked-list > tr
      {
        active-on! : self:checked = !self:checked;
      }
      a#toggler
      {
        click! : $(table#checked-list>tr) -> @(row) row:checked = !row:checked;
      }
      table#checked-list td:nth-child(1)
      {
        padding-left: system-small-icon-width;
        background-repeat:no-repeat;
        background-position:1px 50%;
        background-image:url(theme:check-normal);
      }
      table#checked-list > tr:checked > td:nth-child(1)
      {
        background-image:url(theme:check-checked-normal);
      }
      table#checked-list > tr:hover
      {
        background-color:white white orange orange;
        cursor:pointer;
      }
    </style>
  </head>
<body>
  带有可选项的简单行单元格，底部链接可改变选择状态。
  <table #checked-list border=1px >
    <tr><td>001</td><td>002</td><td>003</td></tr>
    <tr><td>011</td><td>012</td><td>013</td></tr>
    <tr><td>021</td><td>022</td><td>023</td></tr>
    <tr><td>031</td><td>032</td><td>033</td></tr>
    <tr><td>041</td><td>042</td><td>043</td></tr>
  </table>
  <a #toggler href=#>选择状态切换</a>
```

```
</body>
</html>
```

文本宽度测量

效果图


```
<html>
<head>
  <style>
    input[type="text"]
    {
      value-changed!: $1(span#out):value = self.text-width(self:value);
    }
  </style>
<head>
<body>
  文本宽度: <input type="text" /> 为 <span #out></span> 像素
</body>
</html>
```

文本编辑框操作（置选中，取选中字符，取字符）

效果图


```
<html>
<head>
  <style #actions>
    button#to-first
    {
      click!: inp = $1(textarea#test),
              inp.setSelection(0,0),
              inp:focus = true;

    }
    button#to-last
    {
      click!: inp = $1(textarea#test),
```

```
                inp.setSelection(inp:value.length,inp:value.length),
                inp:focus = true;

    }
    button#copy
    {
        click!: inp = $1(textarea#test),
        $1(code#num-chars):value = inp.selectionText();
    }
    button#set-sel
    {
        click!: inp = $1(textarea#test),
                inp.setSelection(0,5);
    }
    button#get-sub
    {
        click!: inp = $1(textarea#test),
                $1(code#num-chars):value = inp:value.substr(12);
    }
    textarea#test
    {
      value-changed!: $1(code#num-chars):value = self:value.length;
    }

    textarea[maxlength]
    {
      value-changed!: maxlength = int(self.maxlength),
                      text = self:value,
                      text.length > maxlength ?
                        ( self:value = text.substr(0,maxlength),
                          self.setSelection(maxlength,maxlength) );
    }
  </style>
<head>
<body>
  <textarea #test cols=80 rows=4>
0123456789
0123456789
  </textarea>
 <button #to-first>|&lt;</button><button #to-last>&gt;|</button>
 <button #set-sel>设置选中</button>
 <button #copy>取选中字符</button>
 <button #get-sub>取字符串</button>
 <text>字符:<code #num-chars /></text>
<p>CSSS! 最大长度处理测试</p>
  <textarea cols=80 rows=4 maxlength=34>
0123456789
```

```
0123456789
  </textarea>
</body>
</html>
```

表格计算

效果图

在右侧列输入数字然后"合计："将计算出列总和

数值 #1	
数值 #2	
数值 #3	
数值 #4	
数值 #5	
数值 #6	
数值 #7	
数值 #8	
数值 #9	
数值 #10	
合计：	0

```
<html>
  <head>
    <style>
      td.number
      {
        width:max-intrinsic;
        behavior:number;
        text-align:right;
        value-changed!:
          total = 0, // 声明变量 total 初始值为 0
          $(td.number) -> @(el) total = total + el:value, // 调用右边的的函数遍历左边的一组满足条件的元素集合 $(td.number)
          $1(td#total):value = total; // 把 td#total 值赋值给 total
      }
      td.number[invalid]
      {
        color:red;
      }
      td#total
      {
        text-align:right;
        font-weight:bold;
      }
      span.comment { color:green; }
    </style>
  </head>
<body>
  <p>在右侧列输入数字然后“合计：”将计算出列总和</p>
  <table border=1>
```

```
    <tr><td>数值 #1</td><td .number>0</td></tr>
    <tr><td>数值 #2</td><td .number>0</td></tr>
    <tr><td>数值 #3</td><td .number>0</td></tr>
    <tr><td>数值 #4</td><td .number>0</td></tr>
    <tr><td>数值 #5</td><td .number>0</td></tr>
    <tr><td>数值 #6</td><td .number>0</td></tr>
    <tr><td>数值 #7</td><td .number>0</td></tr>
    <tr><td>数值 #8</td><td .number>0</td></tr>
    <tr><td>数值 #9</td><td .number>0</td></tr>
    <tr><td>数值 #10</td><td .number>0</td></tr>
    <tr><td>合计：</td><td #total>0</td></tr>
  </table>
</body>
</html>
```

拆分按钮

效果图


```
import win.ui;
/*DSG{{*/
var winform = ..win.form(text="HTMLayout 拆分按钮";right=599;bottom=399)
winform.add()
/*}}*/

import web.layout;
var wbLayout = web.layout( winform );

wbLayout.html = /**
<style>

@set split-button{
//定义混入类，CSS 里特殊语句都会以@字符开始，在混入类里使用:root 表示自己(:root 表示自己在很多地方都有用到)
    :root{
        background-image:url(theme:toolbar-button-hover);
        background-repeat:stretch;
        flow: horizontal;    //使该容器内的块节点(display:block)改变默认的垂直布局为水平布局
        height:min-intrinsic;    //min-intrinsic 指的是节点包含内容的实际大小。也就是文字多大按钮多大
        width:min-intrinsic;
        font:system;
        margin:2px;
    }
```

```
    :root:disabled{
        background-image:url(theme:toolbar-button-disabled);
    }
    :root > caption{
        behavior: clickable;  //该 behavior 使节点可以响应 onButtonClick 事件
        background-image:url(theme:toolbar-split-button-normal);
        background-repeat:stretch; //拉伸图像，theme 返回的图标都是很小的背景图，不位伸看不到效果
        color:windowtext;
        cursor:default;
       white-space: nowrap;
        padding:4px 8px;
        width:*;  //这个写法实际上等价于 width:100%%; 也就是撑满可用的剩余空间
        height:*;
    }
    :root > .dropdown {
        behavior: clickable popup-menu;  //使该节点可以响应 onButtonClick 事件，而且点击该按钮可以弹出子菜单
        background-image:url(theme:toolbar-split-dd-button-normal);
        background-repeat:stretch;
        width:15px;
        height:*;
    }
    :root:hover > caption{
        background-image:url(theme:toolbar-split-button-hover);
    }
    :root:hover > .dropdown {
        background-image:url(theme:toolbar-split-dd-button-hover);
    }
    :root > caption:active {
        background-image:url(theme:toolbar-split-button-pressed);
    }
    :root .dropdown:active  {
        background-image:url(theme:toolbar-split-dd-button-pressed);
    }
    :root:checked > caption {
        background-image:url(theme:toolbar-split-button-checked);
    }
    :root:checked .dropdown  {
        background-image:url(theme:toolbar-split-dd-button-checked);
    }
    :root:checked:hover > caption {
        background-image:url(theme:toolbar-split-button-checked-hover);
    }
    :root:checked:hover .dropdown  {
        background-image:url(theme:toolbar-split-dd-button-checked-hover);
    }
}
widget[type="split-button"],
```

```
input[type="split-button"]{
    style-set:split-button;  //split-button 实际上是一个 CSS 混入类。参见：style-set 重用 CSS 样式
}
</style>

<widget type="split-button" >
    <caption #split-button>拆分按钮</caption>
    <widget .dropdown>
        <menu .popup>
            <li>菜单项一</li>
            <li>菜单项二</li>
        </menu>
    </widget>
</widget>
**/

wbLayout.onButtonClick ={
    ["split-button"] = function (ltTarget,ltEle,reason,behaviorParams) {
        wbLayout.documentElement.insertAdjacentHTML("beforeEnd","点了拆分按钮")
    }
}
wbLayout.onMenuItemClick = function (ltTarget,ltEle,reason,behaviorParams) {
    wbLayout.documentElement.insertAdjacentHTML("beforeEnd","点了菜单:" + ltTarget.innerText )
}

winform.show()
win.loopMessage();
```

五、flow 布局样式

布局方向

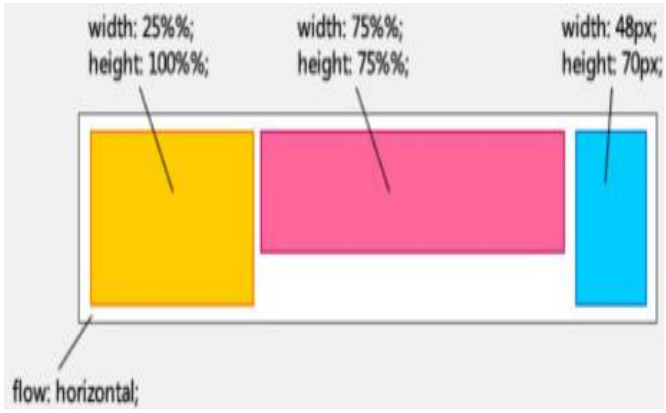
direction: ltr	从左到右布局
direction: rtl	从右到左布局

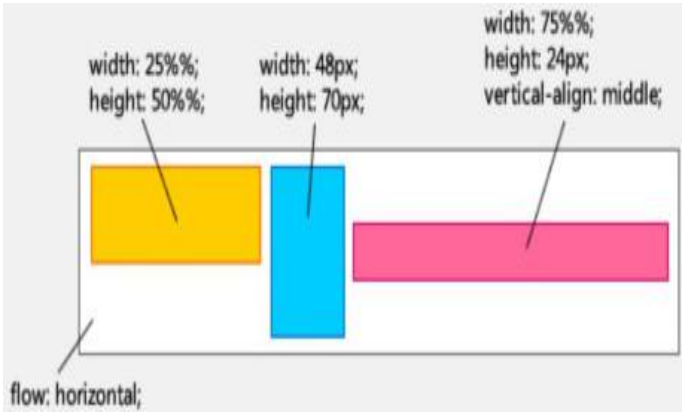
自适应高度和自适应宽度
width: 50%%;
height: 100%%;

使用双百分号表示占用剩余空间的比例
margin 和 padding 等属性也可以使用 %% 单位。100%% 也可以用 * 号表示

垂直/水平布局

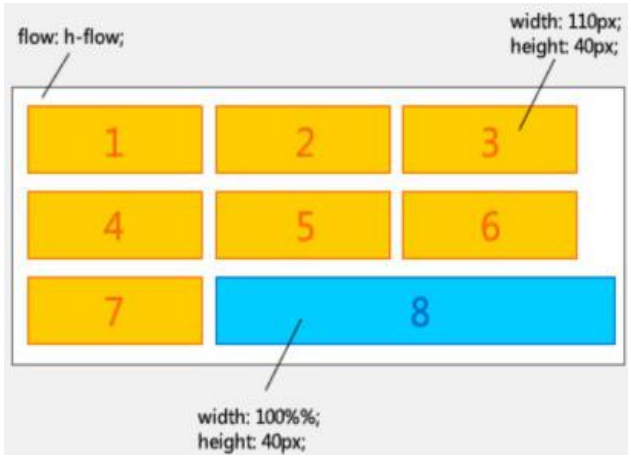
flow: vertical	将容器内部变为垂直布局. (标准的布局模式)
flow: horizontal	将容器内部变为横向布局

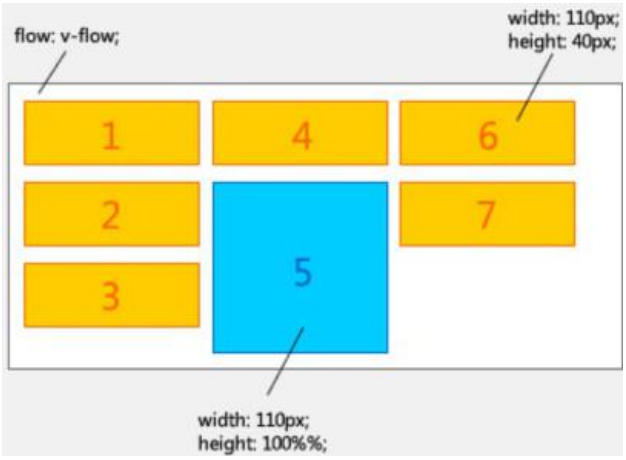




垂直/水平流式布局

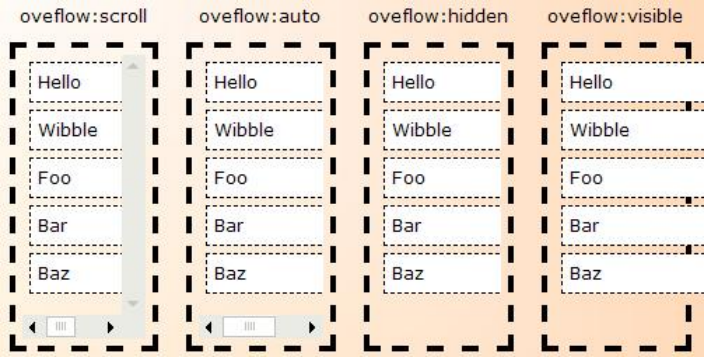
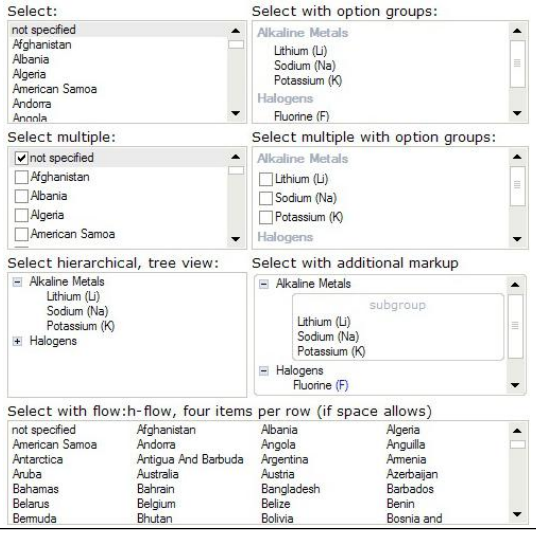
flow: v-flow	将容器内部变为垂直流式布局
flow: h-flow	将容器内部变为水平流式布局





Htmlyout 特有 flow 块容器的布局属性及方法描述，典型的块级容器如：div，ul，ol，blockquote 等，flow 的属性值： vertical，horizontal，h-flow，v-flow，grid

属性值	描述
vertical	默认值，标准的 布局。容器中的所有静态子元素将会被替换为从上到下、一个接一个的根据容器宽度形成一个单一的列 %%单位计算： 宽度，容器中的每个内容区块的左右外边距、 边框和内边距将会使用容器的宽度值%%来计算。在这里容器宽度减去子元素内容的最小宽度得到一个自由的可供分配空间 高度，容器中的所有内容区块的顶部和底部的外边距，边框和内边距（如果给出%%单位）参与自由空间布局。因此，如果容器的内容最小高度小于自由空间的高度。则这个自由空间的所有块部分竞争分布，换句话说，所有块的各部分在%%单位具有限定值
horizontal	这是一个单行布局。 容器的所有静态子元素会水平地一个接一个的排列成一行 %%单位计算： 它类似于 vertical 但方向不同： 高度，容器中的每个内容区块的顶部和底部的外边距，边框和内边距相对于容器给出的%%单位高度来计算。在这里容器高度减去内容块的高度得到可分配的自由空间

	宽度，容器中的所有静态子元素的左右外边距，边框和内边距（如果给出%%单位）参与自由空间布局。因此，如果容器的内容最小高度小于自由空间的高度。则这个自由空间的所有块部分竞争分布，换句话说，所有块的各部分在%%单位具有限定值 示例：  1 个 flow:horizontal 布局容器包含 4 个 overflow 块容器
h-flow	多行水平布局。允许容器的子元素在水平方向上没有足够空间时换行。 %%单位计算： 如果给出%%单位被忽略，vertical 外边距会被解析为未定义。vertical 边框，内边距和高度在%%单位中针对行中最高元素的高度被计算 宽度，行中所有子元素的左右外边距，边框和的内边距完全按照 flow:horizontal 布局计算。 行分裂算法 拼接单行，HTMLayout 试图取代现有水平内容区块。如果满足以下条件之一将分配到新行： 1. 当前区块的宽度，内边距，边框的总和超过可用的容器宽度 2. 该行中的所有%%单位的总和超过 100 第二个条件允许定义 h-flow 布局列表。例如： ul { flow:h-flow; } ul li { width:50%%; } 然后列表项将被放置在两列中，同时每一行元素将超过 ul 元素的可用宽度。 示例： 

v-flow	垂直可换列布局。类似 h-flow 布局，但在垂直方向上，元素会从上到下的排列放置。如果容器没有足够的垂直空间，元素会换列，变成多列布局。
grid	多行多列布局。接近<table>布局，但并不需要<tr>和可以作为一个网格的任意块级元素。 子元素左/顶/右/底部属性，以及用于定义最后一格块位置的特殊的网格位置单位(#)。 例如： #cell { left:2#; top:1#; right:3#; bottom:2#; } 使用 id = “cell”代替表格中从 R1：C2 到 R2：C3 的 4 个网格区块。注意左/顶/右/下属性定义，从 1 开始正向编号网格。 网格区块的宽度和高度属性定义该元素的边界框的尺寸（通常它们是内容框的尺寸）。 %%单位计算： 网格区块的宽度和高度声明使用%%（或*）flex 单位对表格容器（具有 flow:grid 声明）的宽度和高度重新计算。见 sdk/html_samples/flows/grid***.htm 示例。 网格元素的内边距和边框使用 Flex 设定的单位对区块本身的尺寸重新计算。可以这样使用 #cell { left:2#; top:1#; right:3#; bottom:2#; padding-top:*; padding-bottom:*; } 对齐单元块的中间的垂直的内容。 <table><tr><td colspan="2">Top</td></tr><tr><td>Left</td><td>Right</td></tr></table> 例如，在 HTML 和样式声明允许定义类似右边的表格布局： <div #container> <p #left>Left</p><p #right>Right</p> <p #top>Top</p> div#container { flow:grid; } p#left { left:1#; top:2#; } p#right { left:2#; top:2#; } p#top { left:1#; right:2#; top:1#; } 注意：引擎不检查网格重叠。所以多个块可占据同一网格。

HTMLayout - 网格布局

```
import win.ui;
/*DSG{{*/
var winform = ..win.form( bottom=399;parent=...;text="aardio Form";right=599 )
winform.add(  )
/*}}*/

import web.layout;
wbLayout = web.layout(winform)

wbLayout.html = /**
```

```
<body>
<div #container>
<p #a>Left</p><p #b> Right</p>
<p #c>Top</p>
</div>
</body>
**/

wbLayout.css = /**
div#container { flow:grid; }
p#a { left:1#; top:2#; } //ID 为 a 的节点,横坐标序号为 1,纵坐标序号为 2, #可以理解为行号或列号
p#b { left:2#; top:2#; } //ID 为 b 的节点,横坐标序号为 2,纵坐标序号为 2, 也就是在第 2 行第 2 列
p#c { left:1#; right:2#; top:1#; } //ID 为 c 的节点, 左坐标为 1,右坐标为 2,横坐标为 1, 也就是位于第一行的第一列到第二列
**/

winform.show()
win.loopMessage();
```

将左侧想像有一个虚拟的横坐标，用 top 表示一个子节点顶部从哪开始，bottom 表示底部在哪结束。
将顶部想像有一个虚拟的纵坐标, 用 left 表示左侧从哪开始，右侧到哪结束。

HTMLayout - 矩阵布局

```
import win.ui;
/*DSG{{*/
var winform = ..win.form( bottom=399;parent=...;text="aardio Form";right=599 )
winform.add(  )
/*}}*/

import web.layout;
wbLayout = web.layout(winform)

wbLayout.html = /**
<body>
	<div>1.....</div>
	<div>2.....</div>
	<div>3</div>
	<div>4</div>
	<div>5</div>
	<div>6.....</div>
	<div>7</div>
	<div>8</div>
	<div>9</div>
	<div>10</div>
	<div>11</div>
	<div>12</div>
</body>
**/
```

```
wbLayout.css = /**
div { width:*; } //指定宽度才行
body {
  flow: "1 1 1 1" //第一行分为四列,都为 body 的第一个子节点占用
        "2 2 6 6" //第二个子节点占用两列, 第六个节点占用两列
        "7 8 9 10" //其他元素顺序排列
        "11 12 3 4";
  border-spacing-x: 10px;
}

**/

winform.show()
win.loopMessage();
```

简单讲，就是将布局节点内的空间分割成一个一个的格子。
然后每个引号内表示一行,多个字符串表示多行,最后用一个分号结束(css 必须用分号结束一个属性)

然后将节点内的子节点,按他本来的位置,每人分配一个 id（按顺序分给他们 1，2，3，4，5，6，7.....)
然后，谁要占用哪个格子,就在相应位置填上自已的 ID.当然，相同的 ID 必须是连续的。

HTMLayout - 模板布局

```
import win.ui;
/*DSG{{*/
var winform = ..win.form( bottom=399;parent=...;text="aardio Form";right=599 )
winform.add(  )
/*}}*/

import web.layout;
wbLayout = web.layout(winform)

wbLayout.html = /**
<body>
  <div #da> aaaaaa </div>
  <div #db> bbbbbb </div>
  <div #dc> cccccc </div>
</body>
**/

wbLayout.css = /**
body {
  /*布局使用模板名字组成的矩阵,名字放到哪里就表示占用哪个位置*/
  flow: "c a"
        "c b";
}
div{
```

```
  border:1px solid darkgray;
}
div#da{
  float: "a"; //定义一个用于布局的模板名字"a"
  width:50px;
  height:50px;
}
div#db{
  float: "b"; //定义一个用于布局的模板名字"b"
  width:50px;
  height:50px;
}
div#dc{
  float: "c"; //定义一个用于布局的模板名字"c"
  width:50px;
  height:100px;
}
**/

winform.show()
win.loopMessage();
```

flow: <模板表达式> 允许根据模板表达式来替换放置元素。
在这里，模板表达式是一个字符串标识序列。每个字符串标识是一个使用空格分隔的名称标识列表中的一项，其中每个标识指定一个网格中的单元格。多列允许有相同的名称。在这种情况下，该标识相当于定义了一个横跨多个单元网格的占位符。

六、界面贴图技术

1. 概述

在 HTMLayout 中支持在 CSS 中使用 background 属性指定背景图片，并扩展支持更多的功能，例如九宫格切片贴图。

另外，HTMLayout 还可以在 CSS 中使用 foreground 属性指定前景图片。
前景图片的所有属性用法与背景图片完全一样，背景贴图支持的功能，前景贴图同样支持，唯一要做的就是将 background 里的"back"替换为"fore"即可，因为他们用到的语法完全一样，所以在本文中使用星号泛指背景或前景图片，例如 *****ground 在本文中泛指 background 或 foreground。
需要注意的是前景图片显示在 padding 以内（显示区域不包含 padding 部分），而背景图的显示区域则包含 padding 部分

注意：
在 HTMLayout 中 img 图像标记指定的图像被解析为前景图片，img 节点的 src 属性等价于 CSS 中使用 foreground-image 属性。也可以使 用 foreground-**** 修改 img 节点的图像效果。例如 中用 src 属性指定图像等价于在 CSS 中写 foreground-image:url(images/test.jpg) 改变前景图像。

HTMLayout 另外还有一个与 img 类似的 picture 标记，例如：
<picture #photo src="pictures/pic1.jpg" /> 这里的图像也可以使用 foreground-**** 修改样式，但 picture 节点的图像不会被映射为前景图像(foreground-image)，而且 picture 的图像不会进行缓存优化（ 适合用来展示不会频繁被其他节点引用显

示的图片）

可以同时指定背景图片与前景图片，这对于需要大量实现交互效果的软件 UI 设计非常重要。
例如对于一个按钮，他可能有一个前景图标是不会变化的，而他的背景可能需要根据用户的鼠标活动产生动态变化。例如鼠标放到按钮上,离开按钮,按钮按下等等。

HTMLayout 他的优势在于，针对性的对于软件界面的实现提供了很多方便的 CSS 扩展。
而且他的交互响应速度非常快，占用的资源也很少。

2. CSS 标准中的背景属性

首先我们简单回顾一下 CSS 标准语法中与背景图片有关的一些属性:

属性	是否支持	简介
background	支持	背景复合属性
background-color	支持	背景颜色
background-image	支持	背景图像
background-repeat	支持	是否重复排列背景图像
background-attachment	支持	设置背景图像是随内容滚动还是固定的，值为 scroll 时背景随页面其余内容滚动，值为 fixed 时背景不随页面其余内容滚动， HTMLayout 可支持新增的 background-attachment:local; local 指的是节点内部有滚动条时背景随内容滚动（background-attachment:scroll 针对的是外部滚动条）
background-position	支持	设置或检索对象的背景图像位置
background-origin	无	背景绘图起始位置，可选值为 border-box，padding-box，content-box
background-clip	支持	背景显示区域，HTMLayout 支持 border-box，padding-box，不支持 content-box
background-size	支持	背景图像输出大小，HTMLayout 仅在 background-repea:no-repeat 时支持该属性

1、 background 语法:
background: [background-color] || [background-image] || [background-repeat] || [background-attachment] || [background-position]

2、 background-position 语法:
background-position: <bg-position> [, <bg-position>]*
<bg-position> = [<percentage> | <length> | left | center① | right] [<percentage> | <length> | top | center② | bottom]?

设置或检索对象的背景图像位置。必须先指定 background-image 属性。
该属性提供 2 个参数值。 如果提供两个，第一个用于横坐标，第二个用于纵坐标。
如果只提供一个，该值将用于横坐标；纵坐标将默认为 50。这里所指的坐标，指的是图片左上角相对于节点左上角的左标。
例如: background-position 50px 50px; 这可不是指从图片本身的坐标 50x50 开始绘制图片,
而是指将图片向右移动 50 像素，向下移动 50 像素，这里指的坐标是节点容器的坐标系. 而不是图片上的坐标。

默认值：0% 0%，效果等同于 left top

取值:

<percentage>： 用百分比指定背景图像填充的位置。可以为负值。
<length>： 用长度值指定背景图像填充的位置。可以为负值。
left： 背景图像在横向上填充从左边开始。
center①： 背景图像在横向上填充从中间开始。
right： 背景图像在横向上填充从右边开始。
top： 背景图像在纵向上填充从顶部开始。
center②： 背景图像在纵向上填充从中间开始。
bottom： 背景图像在纵向上填充从底部开始。

3. CSS 贴图属性

前景图片一般位于背景图片前面，即使不是同一个节点对象，例如 div 对象#A 包含 div 节点#B,那么#A 的前景图片在#B 的背景图片前面(但是#A 的背景图片仍然在 #B 的背景图片后面)。

下文使用 *****ground 泛指 background 或 foreground

在 HTMLayout 中 可以使用 ****ground-repeat 指定图片的呈现规则，可选值如下:

1、 ****ground-repeat: no-repeat

在这种模式下不进行任何缩放处理，不会重复铺排背景，并且可以配合以下 ****ground-position 属性设置绘图坐标:

****ground-position-left	左侧坐标，指的是图片左上角相对于节点 padding box(就是节点包含 padding 的空间)左侧的偏移值，这个值如果为正数,表示图片左上角向右移动，如果为负数,图片向左移动
****ground-position-top	顶部坐标，指的是图片左上角相对于节点 padding box(就是节点包含 padding 的空间)顶部的偏移值，这个值如果为正数,表示图片左上角向下移动，如果为负数,图片向上移动
****ground-position-right	指的是图片右下角相对于节点 padding box(就是节点包含 padding 的空间)右侧的偏移值，这个值如果为正数,表示图片右下角向左移动，如果为负数,图片向右移动。如果同时指定了 ****ground-position-left，那么 ****ground-position-left 被忽略
****ground-position-bottom	底部坐标，指的是图片右下角相对于节点 padding box(就是节点包含 padding 的空间)底部的偏移值，这个值如果为正数,表示图片右下角向上移动，如果为负数,图片向下移动。如果同时指定了 ****ground-position-top，那么 ****ground-position-top 被忽略
background-position	横坐标 纵坐标；等于同时指定下面的属性: ****ground-position-bottom: 左侧坐标 ****ground-position-bottom: 顶部坐标

2、 ****ground-repeat: repeat

图像重复铺排,填满节点内部空间(包含边框)，可以配合下面的属性使用:

****ground-attachment	可选值为 scroll 或 fixed；设置背景图片是否跟随滚动条滚动，设为 fixed 则固定背景图片不滚动
****ground-position-left ****ground-position-top ****ground-position-right ****ground-position-bottom	定义图片初始编移位置，用法参考 no-repeat

3、 ****ground-repeat: repeat-x

图片横向重复铺排，可以配合下面的属性使用：

****ground-position-right 或 ****ground-position-left	定义开始铺排的横坐标,这两个属性不能同时使用，如果指定 ****ground-position-left 为一个正数则图片向右移动指定的距离然后开始向下铺排，如果指定了 ****ground-position-right 并且是一个正数，则图片向左移动指定的距离，也就是说页面右侧有指定大小的空间没有背景图片
****ground-position-top 或 ****ground-position-bottom	定义图片的顶部偏移,这两个属性不能同时使用

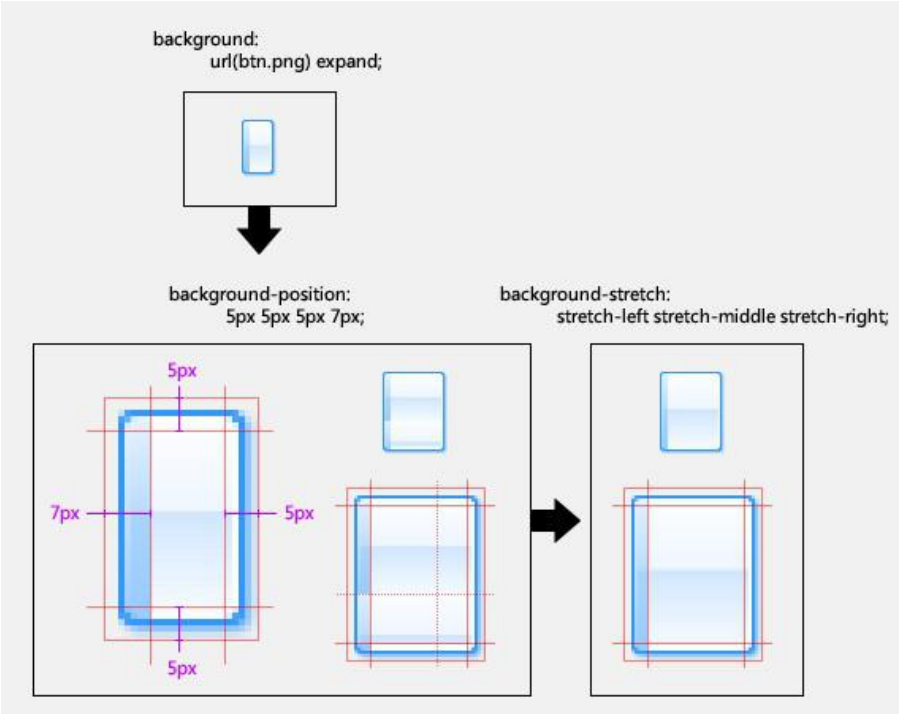
4、 ****ground-repeat: repeat-y

图片垂直重复铺排. 可以配合下面的属性使用：

****ground-position-left 或 ****ground-position-right	定义图片水平偏移,这两个属性不能同时使用
****ground-position-top 或 ****ground-position-bottom	定义开始铺排的纵坐标,这两个属性不能同时使用 如果指定了 ****ground-position-bottom 并且是一个正数，则图片向上移动指定的距离，也就是说页面底侧有指定大小的空间没有背景图片

5、 ****ground-repeat: expand

九宫格切图模式



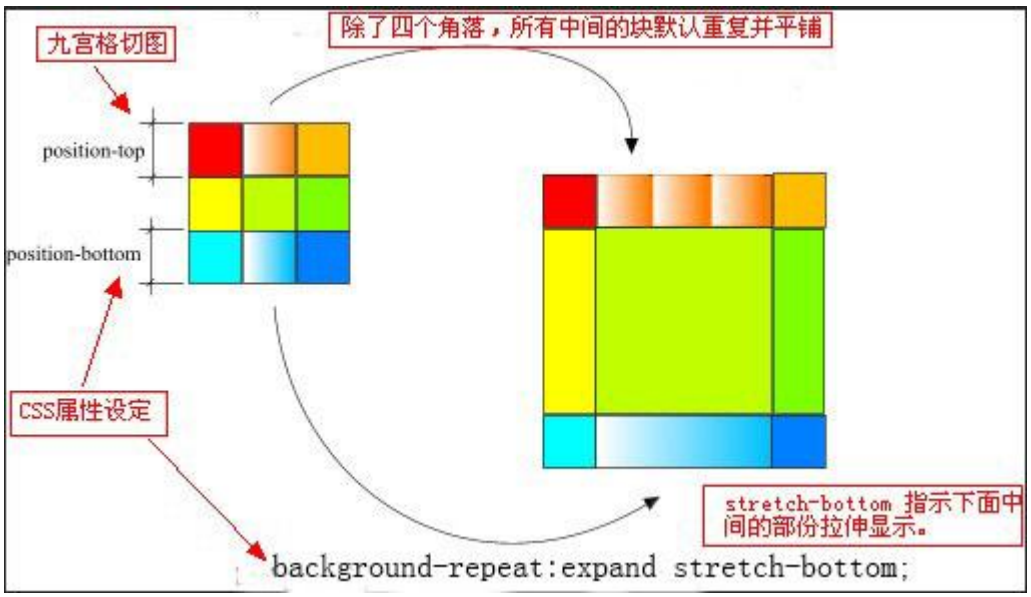
可以配合以下属性指定切片坐标，使图片分为九个分区：

****ground-position-left	左侧块切片位置
****ground-position-top	指定顶部切片位置
****ground-position-right	指定右侧切片位置
****ground-position-bottom	指定底部切片位置
****ground-position	按上,右,下,左的顺序同时指定所有切片位置

示例：

background-position: 6px 6px 6px 6px;
foreground-position: 6px 6px 6px 6px;

请参考下图中的源图片,图片大小为 90 个像素,指定下面的 CSS 进行九宫格切图。
background-position: 30px 30px 30px 30px;



切图后图片如上图分为九个部分，其中四个角落的图片保持原状态放置到节点内部空间(包含 padding 指定的内边距) 四个角上，四角切片不进行任何拉伸或重复铺排。

而其他位于中间部位的图片(顶部中间,底部中间,左侧中间,右侧中间,正中间)，默认都进行重复平铺绘图。

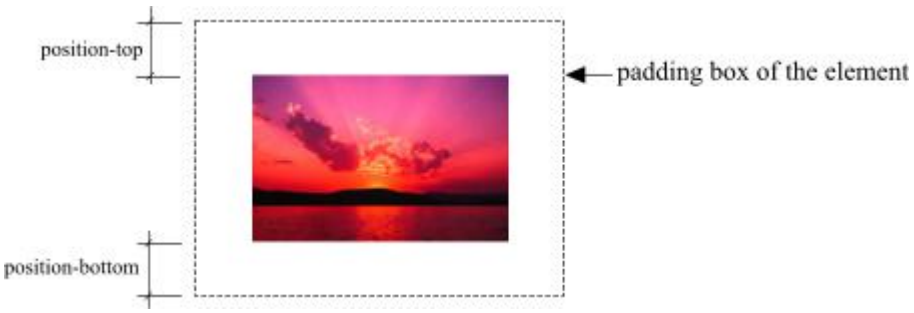
如果需要对这些位于中间部位的图片进行拉伸处理，可以使用 CSS 中的 ****ground-stretch 指定拉伸方式.。

****ground-stretch 可用的参数有:

stretch-left	拉伸左中切块
stretch-middle	拉伸正中切块
stretch-right	拉伸右中切块
stretch-top	拉伸上中切块
stretch-bottom	拉伸下中切块

这些参数也可以追加在 ****ground-repeat 后面，例如：****ground-repeat:expand stretch-bottom;
6、 ****ground-repeat: stretch

图片以拉伸模式显示,即自动适应节点内部空间大小，这里指定内部空间也是包含 padding 内边距的空间(padding box)



可以配合以下 CSS 属性使用:

****ground-position-left: 左边距	指定图片左侧边距，也就是节点左侧指定的大小不显示图片
****ground-position-top: 顶边距;	指定图片顶部边距，也就是节点顶部指定的大小不显示图片
****ground-position-right: 右边距;	指定图片右侧边距，也就是节点右侧指定的大小不显示图片
****ground-position-bottom: 底边距	指定图片底部边距，也就是节点底部指定的大小不显示图片

4. 图像变换效果

HTMLayout 支持对前景和背景图像的多种变换效果，目前可用的变换效果有如下几种：

****ground-image-transformation: opacity(0.5)	设置图像透明度，参数为大于 0 小于 1 的小数．1.0 为不透明，0.0 为完全透明
****ground-image-transformation: flip-x()	图像左右翻转
****ground-image-transformation: flip-y()	图像上下翻转
****ground-image-transformation: colorize(#FFCC00)	颜色滤镜效果，常用于实现按钮图标的禁用状态
****ground-image-transformation: contrast-brightness-gamma(对比度 = 0.0 .. 1.0, 亮度 = 0.0 .. 1.0, gamma 值)	对比度,亮度,gamma 值调整. 对比度和亮度的中间值为 0.5. 可用于鼠标悬停死的按钮图标变化
****ground-image-transformation: color-schema(red,yellow,blue)	将图像的灰度色部分按给出的颜色(可以有多个)进行插值变换

下面是在 CSS 是应用的范例:

CSS	img:hover {
	  foreground-image-transformation: contrast-brightness-gamma(0.5,0.5,1.2);
	}

七、非标准 HTML 标记

1. <include>

此标记允许你将一个外部 HTML 文件作为 HTML 代码片段包含到当前文档,有两种用法：

```
<include src="HTML 片段文件 URL" />

<include src="HTML 片段文件 URL" >
    当读取外部文件失败时,可以显示这里的备选内容
</include>
```

下面是一个简单的示范：
doument-base.html 文件内容如下：

```
<html>
  <body>
    示范文件
    <include src="document-fragment.html">
    <em> 杯具! 打开 document-fragment.html 失败!</em>
  </include>
  </body>
</html>
```

document-fragment.html 内容如下:

```
<P><p>这是 HTML 代码片段.</p>
<p>在这里你可以使用 HTML 代码</p></P>
```

那么 HTMLayout 加载 doument-base.html 以后最终呈现的 HTML 如下

```
<P><html>
  <body>
    示范文件
    <p>这是 HTML 代码片段.</p>
    <p>在这里你可以使用 HTML 代码</p>
    <em> 杯具! 打开 document-fragment.html 失败!</em>
  </include>
  </body>
</html></P>
```

在被包含的文件中,当前路径是包含他的文件所在目录。也就是相对路径是相对于 HTMLayout 当前打开的页面,而不是被包含的文件。储如 href 链接属性,图片 src 属性,嵌入 HTML 的 CSS 样式等都遵守此规则。

2. <widget>

该标记是类似于 <input> <select> 的内联块标记.
<widget> 基本类似 <input>,区别是可以扩展 type 属性,
除了可以使用标准库的类型 (input 节点可以使用的类型,例如: text, checkbox, radio, 等等.)

type 属性也可以定义为下面的值:
<widget type="select"> - 列表框;
<widget type="dropdown-select"> - 下拉列表;
<widget type="textarea"> - 文本框;
<widget type="htmlarea"> - HTML 控件.
<widget> 必须与闭合标记 </widget> 配对使用.

根据 HTML 规范, input 节点会被包含在一个匿名的文本容器中,例如:
<anonymous-text-container><input .../></anonymous-text-container>
而 widget 则不会这么做, 这样使用
%或%%时计算相对高度时不会导致不必要的混淆。
另外 widget 也可以用于实现自定义控件, 在 aardio 中为 widget 节点对象声明 cls,id 等属性即可创建自定义控件

3. <popup>

该标记是 block 样式显示的块标记(类似), 用于定义可在页面上弹出显示的节点, 默认的 popup 是隐藏不显示的。一个常用的功能是使用<popup> 显示复杂些的工具提示(tooltips)

更多参见 HTMLayout 内置 behavior 大全

下面是一个简单的 aardio 范例:

```
import win.ui;
```

```
/*DSG{{*/
var winform = ..win.form( bottom=399;parent=...;text="HTMLayout popup 工具提示";right=599 )
winform.add(  )
/*}}*/
import web.layout;
wbLayout = web.layout(winform)
wbLayout.html = /**
<p titleid="my-html-title">把鼠标放在这里别动哦，然后可以看到提示</p>
<popup id="my-html-title">
  这是一个 <em>示范</em>:<br/>
可以使用 html 的工具提示 tooltip
</popup>
**/
winform.show()
win.loopMessage();
```

八、HTMLayout 的 CSS

1. style-set 重用 CSS 样式

style-set 类似 SASS、LESS 这些 CSS 预处理语言中的 Mixins(混合)

- 分离出需要重用的 CSS 样式并放到单独的 mixin 里(在 HTMLayout 里称之为 style-set)。

在 CSS 中定义 style-set 的语法:

```
@set 样式组名字 {
    CSS 选择器 {
    }
}
```

使用 style-set 的语法:

```
CSS 选择器 {
    style-set : 样式组名字;
}
```

在 style-set 中可以使用子节点的选择器指定样式。
但是可惜的是不能象 SASS、LESS 那样支持多层嵌套，也不能直接包含根节点的样式 - 根节点的样式必须写在 :root 选择器下面。

示例:

```
import win.ui;
/*DSG{{*/
winform = win.form( right=599;bottom=399;text="style-set 演示")
/*}}*/

import web.layout;
var wbLayout = web.layout( winform );
```

```
wbLayout.html =/***
<html theme="test">
<body>
        <div data-url="http://bbs.aardio.com" >
                <span .cls-a> test 1</span>
                <span .cls-b> test 2</span>
        </div>
        <span .cls-a> test 3 - 注意 style-set 作用域外不受影响</span>
</body>
***/

wbLayout.css = /**
@set css-rule {
        .cls-a{
            color:red;
        }
        .cls-b{
            color:blue;
        }
        :root { /* 使用:root 表示当前作用域的根节点 */
            width:300px;
            height:300px;
            border:1px solid red;
        }
}
html[theme="test"] div[data-url="http://bbs.aardio.com"]{
    style-set:css-rule;
}
**/

winform.show()
win.loopMessage();
```

在 style-set 内部使用伪类 :root 表示当前指定的根节点。

style-set 还有那么一点 OOP 的功能，可以实现简单的继承，如下：

```
@set css-rule-base {
        :root { /* 使用:root 表示当前作用域的根节点 */
            width:300px;
            height:300px;
            border:1px solid red;
        }
}
/*一个尖括号后面指定基类*/
@set css-rule < css-rule-base {
        .cls-a{
        color:red;
```

```
        }
        .cls-b{
          color:blue;
        }
}
```

2. htmlayout 的默认 CSS

HTMLayout 窗口默认加载 MASTER.CSS，里面包含 HTMLayout 自带控件的默认样式，
如果需要修改控件的默认样式可以打开 MASTER.CSS 参考下原来的写法。

```
html { behavior: accesskeys; }
form { behavior:form; }

/* hyperlinks */
a:link { text-decoration:underline; color:blue; cursor:pointer; beha
vior:hyperlink; }
a:link:focus { text-decoration:double-underline; }
a:link:hover { color:red; }
a:link:active { text-decoration:none; }
a:link:tab-focus { outline-width:1px; outline-offset:0px; outline-styl
e:dotted; }

*:drop-marker { visibility:hidden; }

//popup[role="overflow-tooltip"] { white-space:nowrap; overflow:
none; width:max-intrinsic; height:max-intrinsic;}
//popup[role="tooltip"] { white-space:nowrap; overflow:none; wid
th:max-intrinsic; height:max-intrinsic;}

/* edit box and numeric input*/

/* widget { color:windowtext; } */

input[type="hidden"]
{

  display:none;
}

fieldset
{
  background-image:url(theme:groupbox-normal);
  background-repeat:stretch;
}

fieldset > legend
{
  background-color:threedhighlight;
  padding: 2px 4px;
  margin-left:4px;
  margin-right:*;
}
fieldset > legend:rtl /* see http://terrainformatica.com/forums/to
pic.php?id=1772 */
{
  margin-left:*;
  margin-right:4px;
}

widget[type="text"],
input[type="text"],
widget[type="password"],
input[type="password"],
widget[type="currency"],
input[type="currency"],
widget[type="url"],
input[type="url"],
input:not([type])
{
  style-set: "std-edit";
}

@set std-edit
  {
    :root
    {
      font:system;
      color:windowtext;
      padding:3px;
      height:1em;
      width:1em;
      background-image:url(theme:edit-normal);
      background-repeat:stretch;
      min-width: min-intrinsic;
      min-height: min-intrinsic;
      text-align: left;
      cursor:text;
      overflow:hidden;
      white-space: nowrap;
      behavior:edit;
      context-menu:url(res:behavior-edit-menu.htm);
      text-selection: highlighttext highlight;
    }
    :root:disabled
    {
      background-image:url(theme:edit-disabled);
```

```
        color:graytext;
      /*cursor:default;*/
    }
    :root[type="password"]
    {
      behavior:password;
    }
    :root[type="currency"]
    {
      text-align: right;
      behavior:currency;
    }
    :root[type="url"]
    {
      behavior:url;
    }

    :root[invalid]
    {
      color:red;
    }
  }

widget[type="decimal"],
input[type="decimal"],
widget[type="number"],
input[type="number"]
{
      style-set: "std-number-edit";
}

@set std-number-edit
  {
    :root
    {
      text-align: right;
      behavior:number;
      font:system;
      color:windowtext;
          padding:3px;
      height:1em;
      width:1em;
      background-image:url(theme:edit-normal);
      background-repeat:stretch;
      min-width: min-intrinsic;
      min-height: min-intrinsic;
      cursor:text;
      overflow:visible;
      white-space: nowrap;
          flow:vertical;
          context-menu:url(res:behavior-edit-menu.htm);
    }
    :root[type="decimal"]
    {
      behavior:decimal;
      width:2em;
          context-menu:none;
    }

    :root:disabled
    {
      background-image:url(theme:edit-disabled);
          color:graytext;
      /*cursor:default;*/
    }

    :root[invalid]
    {
      color:red;
    }

    :root > button
    {
      behavior:clickable;
      cursor:default;
    }

    :root > button.minus
    {
      padding:0;
      margin: 0 -2px -2px *;
      background-image:url(theme:v-spin-minus-normal);
      background-repeat:stretch;
      width:system-scrollbar-width;
          height:50%%;
    }

    :root:rtl > button.minus
    {
      margin: 0 * -2px -2px;
    }

    :root > button.minus:hover
    {
      transition:none;
      background-image:url(theme:v-spin-minus-hover);
    }
    :root > button.minus:active
    {
      background-image:url(theme:v-spin-minus-pressed);
    }
    :root > button.minus:disabled
    {
      background-image:url(theme:v-spin-minus-disabled);
    }
    :root > button.plus
    {
      padding:0;
      //margin: system-border-width system-border-width 0 *;
          //margin-left:*;
          margin: -2px -2px -1 *;
      background-image:url(theme:v-spin-plus-normal);
      background-repeat:stretch;
          width:system-scrollbar-width;
```

```
            height:50%%;
    }
    :root:rtl > button.plus
    {
      margin: -2px * -1px -2px;
      //border: 1px solid red;
    }

    :root > button.plus:hover
    {
            transition:none;
      background-image:url(theme:v-spin-plus-hover);
    }
    :root > button.plus:active
    {
      background-image:url(theme:v-spin-plus-pressed);
    }
    :root > button.plus:disabled
    {
      background-image:url(theme:v-spin-plus-disabled);
    }

    @media handheld
        {
          :root
          {
                padding:3px;
                flow:horizontal;
          }
          :root > button.plus
          {
                height:100%%;
                margin: -3px 0px -3px *;
          }
          :root > button.minus
          {
                height:100%%;
                margin: -3px -3px -3px -1px;
          }
    }

  }


/* text button */

widget[type="button"],
input[type="button"],
input[type="reset"],
input[type="submit"],
button
{
       style-set: "std-button";
}

@set std-button
  {
:root
{
        font:system;
  color:windowtext;
  padding:4px 8px;
  white-space: nowrap;
  behavior:button;
  background-repeat:stretch;
  background-image:url(theme:button-normal);
  cursor:default;
  text-align:center;
}

:root[role="default-button"]
{
  background-image:url(theme:button-defaulted);
}
:root:hover
{
  background-image:url(theme:button-hover);
  transition: blend;
}
:root:focus
{
  background-image:url(theme:button-defaulted);
}
:root:active
{
  background-image:url(theme:button-pressed);
}
:root:active > *
{
  position:relative;
  left:1px;
  top:1px;
}

:root:disabled
{
  background-image:url(theme:button-disabled);
       color:graytext;
}
:root:tab-focus
{
  //outline:1px dotted invert -4px; /* inside */
       outline:1px dotted -4px; /* inside */
}

/* vertical alignment */
:root > :first-child { margin-top:*; }
:root > :last-child { margin-bottom:*; }

@media handheld
{
     :root:tab-focus
     {
       outline:1px solid -2px highlight;
```

```
        }
        :root:hover
        {
            transition:none;
            color:highlighttext;
        }
    }
  }

widget[type="menu"],
button[type="menu"]
{
    style-set: "std-button-menu";
}

@set std-button-menu < std-button
  {
    :root
    {
      behavior:button popup-menu;
    }
    :root:owns-popup /* button when popup is shown */
    {
      background-image:url(theme:button-pressed); /* ?? */
    }
  }

button[type="selector"]
{
  style-set: "std-selector";
}


/* radio button */
input[type="radio"],
button[type="radio"],
widget[type="radio"]
{
    style-set:"std-radio";
}

@set std-radio
  {
    input:root
    {
      padding:0px;
      height:system-small-icon-height;
      width:system-small-icon-width;
        vertical-align:middle;
      behavior:radio;
      background-repeat:no-repeat;
      background-position:50% 50%;
      background-image:url(theme:radio-normal);
      cursor:default;
    }
      :root:not(input)
    {
      padding:4px 4px 4px system-small-icon-width;
      behavior:radio;
      background-repeat:no-repeat;
      background-position:1px 50%;
      background-image:url(theme:radio-normal);
      //text-decoration:underline;
          vertical-align:baseline;
          text-align:left;
          white-space:nowrap;
      cursor:pointer;
    }
    :root:not(input):rtl
    {
      padding:4px system-small-icon-width 4px 4px;
      background-position-right:1px;
          background-position-left:auto;
    }

    :root:hover
    {
      transition:blend;
      background-image:url(theme:radio-hover);
    }
    :root:active
    {
      background-image:url(theme:radio-pressed);
    }
    :root:disabled
    {
      background-image:url(theme:radio-disabled);
          color:graytext;
    }
    :root:checked
    {
      background-image:url(theme:radio-checked-normal);
    }
    :root:checked:hover
    {
      background-image:url(theme:radio-checked-hover);
    }
    :root:checked:active
    {
      background-image:url(theme:radio-checked-pressed);
    }
    :root:checked:disabled
    {
      background-image:url(theme:radio-checked-disabled);
          color:graytext;
    }
    :root:tab-focus
    {
      outline:1px dotted invert -1px; /* inside */
    }
    @media handheld
    {
        :root:tab-focus
        {
```

```
            outline:1px solid -1px highlight;
        }
    }
  }

/* checkbox button */
input[type="checkbox"],
widget[type="checkbox"],
button[type="checkbox"]
{
      style-set: "std-checkbox";
}

@set std-checkbox
  {
    input:root
    {
      padding:0px;
      height:system-small-icon-height;
      width:system-small-icon-width;
      behavior:check;
      background-repeat:no-repeat;
      background-position:50% 50%;
      background-image:url(theme:check-normal);
      cursor:default;
          vertical-align:middle;
    }
      :root:not(input)
    {
      padding:4px 4px 4px system-small-icon-width;
      behavior:check;
      background-repeat:no-repeat;
      background-position:1px 50%;
      background-image:url(theme:check-normal);
      //text-decoration:underline;
      cursor:pointer;
      white-space:nowrap;
      style-set: "std-checkbox";
          vertical-align:baseline;
    }
        :root:not(input):rtl
    {
      padding:4px system-small-icon-width 4px 4px;
      background-position-right:1px;
          background-position-left:auto;
    }
    :root:hover
    {
      transition:blend;
      background-image:url(theme:check-hover);
    }
    :root:active
    {
      background-image:url(theme:check-pressed);
    }
    :root:disabled
    {
      background-image:url(theme:check-disabled);
          color:graytext;
    }
    :root:checked
    {
      background-image:url(theme:check-checked-normal);
    }
    :root:checked:hover
    {
      background-image:url(theme:check-checked-hover);
    }
    :root:checked:active
    {
      background-image:url(theme:check-checked-pressed);
    }
    :root:checked:disabled
    {
      background-image:url(theme:check-checked-disabled);
        color:graytext;
    }

    :root[mixed]:empty
    {
      background-image:url(theme:check-mixed-normal);
    }
    :root[mixed]:empty:hover
    {
      background-image:url(theme:check-mixed-hover);
    }
    :root[mixed]:empty:active
    {
      background-image:url(theme:check-mixed-pressed);
    }
    :root[mixed]:empty:disabled
    {
      background-image:url(theme:check-mixed-disabled);
          color:graytext;
    }

    :root:tab-focus
    {
      outline:1px dotted invert -1px;
    }
    @media handheld
    {
        :root:tab-focus
        {
            outline:1px solid -1px highlight;
        }
            widget:root,
        button:root
        {
            text-decoration:none;
        }
    }
}
```

```
popup
{
  style-set: "std-popup"; /* used to break chain of style-sets so
<button><popup><button/></popup></button> will work */
}

@set std-popup
{
  /* empty  */
}

button[type="selector"]
{
    style-set: "std-selector";
}

@set std-selector
  {
    :root
    {
      font:system;
      color:windowtext;
      white-space: nowrap;
      background-repeat:stretch;
      background-image:url(theme:button-normal);
      cursor:default;
      text-align:center;
      padding:4px 14px 4px 8px; /* + room for the arrow */
      behavior:popup-selector;
      foreground-image:url(stock:arrow-down); /* that arrow */
      foreground-repeat: no-repeat;
      foreground-position-top: 50%;
      foreground-position-right: 3px;
      overflow: hidden;
      height:min-intrinsic;
    }

    :root > caption { margin-top:*; margin-bottom:*; }

    :root:hover
    {
      background-image:url(theme:button-hover);
      transition: blend;
    }
    :root:focus
    {
      background-image:url(theme:button-defaulted);
    }
    :root:active
    {
      background-image:url(theme:button-pressed);
    }

    :root:active > caption
    {
      position:relative;
          left:1px;
          top:1px;
    }
    :root:disabled
    {
      background-image:url(theme:button-disabled);
          color:graytext;
    }
    :root:tab-focus
    {
          outline:1px dotted -4px; /* inside */
    }
    :root:active
    {
      foreground-position-top: 55%;
      foreground-position-right: 2px;
    }
    :root > menu,
    :root > popup
    {
      style-set: "std-popup-menu";
      display:block;
      visibility:collapse;/* invisible normally */
      border:none;
      min-width: max-intrinsic;
          width: *;
      height: max-intrinsic;
      transition:window-slide-ttb;
    }
    :root > menu:popup,
    :root > popup:popup
    {
      visibility:visible;/* visible when popup */
    }

  }

progress,
input[type="progress"],
widget[type="progress"]
{
  behavior:progress;
  background-repeat:stretch;
  background-image:url(theme:h-progress-back);

  height: 1.2em;
  width: 8em;
  padding: 3px;

  foreground-repeat:stretch;
  foreground-image:url(theme:h-progress-chunk);
}

/* select (not dropdown) */

select[type="select"],
widget[type="select"],
popup[type="select"]
```

```
{
  style-set: "std-select";
}

@set std-select
{
        :root
        {
          font:system;
                behavior:select;
                overflow-x:hidden;
                overflow-y:auto;
                flow:vertical;
                padding:2px;
                background-repeat:stretch;
                background-image:url(theme:edit-normal);
                width: max-intrinsic;
                height: min-intrinsic;
        }

        :root:disabled
        {
                //overflow:hidden;
                background-image:url(theme:edit-disabled);
                color:graytext;
        }

        optgroup > caption
        {
                padding:3px 3px;
                font-weight:bold;
                color: graytext;
        }

        option
        {
                padding:1px 1px 1px 3px;
                color: windowtext;
                min-height:1em;
                width: 100%%;
                white-space: nowrap;
        }
        option:rtl
        {
                padding:1px 3px 1px 1px;
        }

        option:checked:not(:has-children-of-type(option))
        {
                background-color:threedface; color:windowtext;
        }

        :root:focus option:current:not(:has-children-of-type(option)),
        popup:root option:current:not(:has-children-of-type(option))
        {
                background-color:highlight; color:highlighttext;
        }

          optgroup > option
          {
                  padding-left:19px;
          }
          optgroup > option:rtl
          {
                  padding-right:19px;
                  padding-left:0;
          }

        option:has-children-of-type(option),
          options
          {
                  padding-left:32px;
                  foreground-repeat: no-repeat;
                  foreground-position: 3px 3px;
                  width: 100%%;
                  min-height:1em;
                  foreground-image-cursor:pointer;
          }
          option:has-children-of-type(option):rtl,
          options:rtl
          {
            padding-right:32px;
                  padding-left:0;
                  foreground-position-right: 3px;
                  foreground-position-left: auto;
          }

          option:has-children-of-type(option):collapsed,
          options:has-children:collapsed {    foreground-image:url(the
me:tree-view-glyph-closed); }

        option:has-children-of-type(option):expanded,
          options:has-children:expanded  { foreground-image:url(the
me:tree-view-glyph-open); }

          /* all elements in the options by default are non-visible */
        option:has-children-of-type(option) > *,
          options > *       { visibility:collapse; }

          /* elements in open options are visible */
          option:has-children-of-type(option):expanded > *,
          options:expanded > * { visibility:visible; }

          /* options caption element, always visible */
          option:has-children-of-type(option) > :first-child,
          options > :first-child
          {
            padding:1px 1px 1px 3px;
            visibility:visible;
            width: 100%%;
            white-space: nowrap;
            margin-left:-16px;
          }
          option:has-children-of-type(option) > :first-child:rtl,
```

```
        options > :first-child:rtl
        {
          padding:1px 3px 1px 1px;
          margin-right:-16px;
          margin-left:0;
        }
        option:has-children-of-type(option):current > :first-child,
        options:current > :first-child
        {
          background-color:threedlight; color:windowtext;
        }

    :root:focus option:has-children-of-type(option):current > :first-ch
ild,
        :root:focus options:current > :first-child
        {
          background-color:highlight; color:highlighttext;
        }
}

select[type="tree"],
widget[type="tree"]
{
    style-set: "std-tree";
}
select[type="tree"][checkmarks],
widget[type="tree"][checkmarks]
{
    style-set: "std-tree-checkmarks";
}

@set std-tree
{
        :root
        {
          font:system;
          behavior: tree;
              overflow-x:hidden;
              overflow-y:auto;
              flow:vertical;
              padding:2px;
              background-repeat:stretch;
              background-image:url(theme:edit-normal);
              width: max-intrinsic;
              height: min-intrinsic;
        }

        :root:disabled
        {
              background-image:url(theme:edit-disabled);
              color:graytext;
        }

        option
        {
              color: windowtext;
              min-height:1em;
              width: 100%%;
              white-space: nowrap;
        }

        optgroup > caption
        {
              padding:3px 3px;
              font-weight:bold;
              color: graytext;
        }

        optgroup > option
        {
              padding-left:19px;
        }
        optgroup > option:rtl
        {
              padding-right:19px;
              padding-left:0;
        }

    option:has-children-of-type(option)
        {
              padding-left:20px;
              foreground-repeat: no-repeat;
              foreground-position: 3px 4px;
              width: 100%%;
              min-height:1em;
              foreground-image-cursor:pointer;
        }
        option:has-children-of-type(option):rtl
        {
          padding-right:20px;
              padding-left:0;
              foreground-position-right: 3px;
              foreground-position-left: auto;
        }

    /* all option captions */
        option > :first-child
        {
          padding:2px 3px 2px 3px;
        }
        option > :first-child:rtl
        {
          padding:2px 3px 2px 3px;
        }

        option:has-children-of-type(option):collapsed
        {       foreground-image:url(theme:tree-view-glyph-closed); }

    option:has-children-of-type(option):expanded
        {       foreground-image:url(theme:tree-view-glyph-open); }

        /* all elements in the options by default are non-visible */
    option:has-children-of-type(option) > *
    { visibility:collapse; }
```

```
    /* elements in open options are visible */
    option:has-children-of-type(option):expanded > *
    { visibility:visible; }

    /* options caption element, always visible */
  option:has-children-of-type(option) > :first-child
  {
      visibility:visible;
      width: 100%%;
      white-space: nowrap;
    margin-left:-6px;
  }

  option:has-children-of-type(option) > :first-child:rtl
  {
    margin-left:0;
    margin-right:-6px;
  }
  /* :current hightlighting */
     option:current > :first-child
     {
       background-color:threedlight; color:windowtext;
     }
     :root:focus option:current > :first-child,
     popup:root option:current > :first-child
     {
          background-color:highlight; color:highlighttext;
     }

  /* tree line support: */
  :root[treelines] option:has-children-of-type(option) > option
  {
    display: list-item;
    list-style-type: tree-line;
    list-marker-size:1px;
    list-marker-color:threedshadow;
    list-marker-style:dotted;
  }

  option,
  option > :first-child
  {
    width:max-intrinsic;
  }
  //option > :first-child:hover /* highlight caption */
  //{
  //  outline:1px solid highlight -1px;
  //}
}

@set std-tree-checkmarks < std-tree
{
  :root { behavior: tree-checkmarks; }
  option > :first-child /* option caption gets checkmark icon */
  {
    //margin:0;
    padding-left:16px;
    foreground-image:url(theme:check-normal);
    foreground-repeat: no-repeat;
    foreground-position: 2px 50%;
    foreground-image-cursor: pointer;
  }

  option > :first-child:rtl
  {
    padding-left:0;
    padding-right:16px;
    foreground-position-left:auto;
    foreground-position-right:2px;
  }

  option:has-children-of-type(option)
  {
    padding-left:26px;
  }
  option:has-children-of-type(option) > :first-child
  {
    margin-left:-10px;
  }
  option:has-children-of-type(option):rtl
  {
    padding-left:0;
    padding-right:26px;
  }
  option:has-children-of-type(option) > :first-child:rtl
  {
    margin-left:0;
    margin-right:-10px;
  }
  option:incomplete > :first-child { foreground-image:url(theme:c
heck-mixed-normal);  }
  option:checked > :first-child    { foreground-image:url(theme:c
heck-checked-normal); }
}

input[type="file-path"],
widget[type="file-path"],
input[type="folder-path"],
widget[type="folder-path"]
{
  style-set: "std-file-selector";
  behavior:path-select;
}

input[type="file"],
widget[type="file"]
{
  style-set: "std-file-selector";
  behavior:file;
}
@set std-file-selector
{
  :root
```

```
  {
    padding:0;
    font:system;
    flow:horizontal;
    width:20em;
    background-repeat:stretch;
    background-image:url(theme:edit-normal);
  }
  :root:disabled
  {
    background-image:url(theme:edit-disabled);
  }
  :root > caption
  {
    behavior:file-icon;
    width:*;
    padding:4px 4px 4px system-small-icon-width;
    height:*;
    overflow-x: hidden;
    text-overflow:ellipsis;
    foreground-repeat:no-repeat;
    foreground-position: 2px 50%;
  }
  :root:empty > caption
  {
    color:graytext;
  }
  :root:disabled > caption
  {
    color:graytext;
  }
  :root > button
  {
    margin:1px;
    padding:3px 6px;
    width:max-intrinsic;
    height:*;
  }

}

/* simple multiselect (without check boxes) */

select[type="select"][multiple=""],
widget[type="select"][multiple=""]
{
  style-set: "std-select-multiple";
}

@set std-select-multiple < std-select
  {
    :root
    {
      background-color:transparent; color:windowtext;
    }
    option
    {
      padding:1px 1px 1px 3px;
      padding-left:1.6em;
      background-color:transparent; color:windowtext;
    }
    option:rtl
    {
      padding:1px 3px 1px 1px;
      padding-right:1.6em;
      background-color:transparent; color:windowtext;
    }
    option:checked
    {
      list-style-type:disc;
      display:list-item;
      background-color:threedface; color:windowtext;
    }
    :root:focus option:current
    {
      background-color:highlight; color:highlighttext;
    }
  }

/* multiselect with check boxes */

select[type="select"][multiple="checks"],
widget[type="select"][multiple="checks"]
{
      style-set: "std-select-multiple-checkmarks";
}

@set std-select-multiple-checkmarks < std-select
  {
    option
    {
      padding: 1px 1px 1px system-small-icon-width;
      foreground-repeat:no-repeat;
      foreground-position:1px 50%;
      foreground-image:url(theme:check-normal);
      background-color:transparent; color:windowtext;
    }
    option:checked
    {
      foreground-image:url(theme:check-checked-normal);
      background-color:transparent; color:windowtext;
    }
    option:rtl
    {
      padding: 1px system-small-icon-width 1px 1px;
      foreground-position-right:1px;
      foreground-position-left:auto;
    }
  }


/* dropdown combobox */
widget[type="select-dropdown"],
select[type="select-dropdown"]
```

```
{
      style-set: "std-select-dropdown";
}

@set std-select-dropdown
  {
    :root
    {
          font:system;
      behavior:dropdown-select;
      padding:0;
      flow:horizontal;  /* caption, button*/
      background-repeat:stretch;
      background-image:url(theme:edit-normal);
      width: max-intrinsic;
      height: max-intrinsic;
      overflow-x:hidden;
    }

    :root:disabled
    {
      background-image:url(theme:edit-disabled);
          color:graytext;
    }

    /* caption portion of combobox */
    :root > caption
    {
      padding:1px;
      margin:2px 1px 2px 2px;
      /*
          overflow:none;
          min-width:max-intrinsic;
      */
      overflow-x:hidden;
      width:100%%; /* everything left from the button */
      height:100%%; /* takes full height */
          min-height:1em;
          min-width: system-scrollbar-width;
      vertical-align:middle;
      white-space: nowrap;
      /*background-color:window;*/
      color:windowtext;
       }

    /* caption portion of combobox when select is in focus */
    :root:focus > caption
    {
      background-color:highlight; color:highlighttext;
    }

    /* popup select element */
    :root > popup
    {
      overflow-x:hidden;
      overflow-y:auto;
      padding:2px;
      display:block;
      visibility:collapse;/* invisible normally */
      border:none;
      behavior:select;
      min-width: max-intrinsic;
          width: *;
      height: min-intrinsic;
      transition:window-slide-ttb;
    }
    :root > popup:popup
    {
      visibility:visible;/* visible when popup */
    }

        :root > popup option:checked
        {
              background-color:threedface; color:windowtext;
    }

        :root > popup option:current
        {
          background-color:highlight; color:highlighttext;
    }

    /* dropdown button of the combobox */
    :root > button
    {
      width       :system-scrollbar-width;
      height      :*;
      margin      :system-border-width; /*1px 1px 1px 0px;*/
      padding     :0;
      background-repeat     :stretch;
      background-image      :url(theme:combobox-button-normal);
      behavior          :clickable;
      white-space       :nowrap;
    }

    :root > button:hover
    {
          transition :none;
      background-image:url(theme:combobox-button-hover);
    }

    :root > button:active
    {
      background-image:url(theme:combobox-button-pressed);
    }

    :root:disabled > button
    {
      background-image:url(theme:combobox-button-disabled);
        color:graytext;
    }

    :root[editable] > caption
    {
      behavior:edit;
```

```
        cursor:text;
            context-menu:url(res:behavior-edit-menu.htm);
    }
    /*
    @media ms-vista-aero
    {
      :root   { background-image:url(theme:button-normal); }
      :root:hover { background-image:url(theme:button-hover);   }
      :root:disabled { background-image:url(theme:button-disable
d); }
      :root:active { background-image:url(theme:button-pressed);
 }
      :root > caption { padding: 2px; background-color:transpare
nt; color:windowtext; }
      :root:focus > caption { outline:1px dotted -2px; }
      :root > button { background: url(stock:arrow-down) center
center no-repeat;}
      :root > button:hover { background-image:url(stock:arrow-d
own); background-position: center center; background-repeat: no
-repeat;}
      :root > button:active { background-image:url(stock:arrow-d
own); background-position: center center; }
      :root:disabled > button   { background-image:url(stock:arro
w-down); background-position: center center; }
    }
    */
  }

@set std-richtext
{
  :root
{
    flow:null;
    font: 10pt Verdana, Arial, sans-serif;
    width:*;
      height:150px;
      padding:3px;
      background-image:url(theme:edit-normal);
      background-repeat:stretch;
      overflow-x:auto;
      overflow-y:scroll;
        //cursor:text;
      behavior:richtext;
      context-menu:url(res:behavior-richtext-menu.htm);
        text-selection: windowtext threedface;
  }
  :root:focus
  {
    text-selection: highlighttext highlight;
  }
      :root:empty { color:graytext; vertical-align:top;}
}

@set std-textarea
{
    :root
{
        flow:null;
        padding:system-3d-border-width;
  background-image:url(theme:edit-normal);
  background-repeat:stretch;
        width:*;
        height:*;
  overflow-x:auto;
  overflow-y:scroll;
        //cursor:text;
  white-space:prewrap;
  font: system;
  behavior:plaintext;
  context-menu:url(res:behavior-text-menu.htm);
        text-selection: windowtext threedface;
      }

  :root:focus
  {
    text-selection: highlighttext highlight;
}

      :root:disabled
      {
            background-image:url(theme:edit-disabled);
            overflow:hidden;
            cursor:default;
    }
      textarea:root
      {
          min-width: min-intrinsic;
          min-height: min-intrinsic;
          width: min-intrinsic;
          height: min-intrinsic;
      }
      plaintext:root
      {
          width: *;
        height: *;
      }
      :root[wrap="off"]
{
    overflow-x:scroll;
    overflow-y:scroll;
    white-space:pre;
}
      :root:empty { color:graytext;}
      plaintext:root:empty { color:graytext; vertical-align:top;}
}

textarea, widget[type="textarea"], plaintext
{
  style-set: "std-textarea";
}

richtext, widget[type="richtext"]
{
  style-set: "std-richtext";
```

```
}

textarea:disabled, richtext:disabled,
widget[type="textarea"]:disabled, widget[type="richtext"]:disabled,
plaintext:disabled
{
        background-image:url(theme:edit-disabled);
      overflow:hidden;
        cursor:default;
}




popup
{
      border:1px solid threedshadow;
        background-color: infobackground;
        overflow:none;
        text-overflow:none;
        //font:system;
        color:infotext;
        padding:1px 2px;
        display:none; /* invisible normally */
        width: max-intrinsic;
        height: max-intrinsic;
}

widget[type="hslider"],
input[type="hslider"]
{
        style-set:"std-hslider";
}

@set std-hslider
{
     :root
        {
             behavior:slider;
               cursor:default;
               padding:2px;
               height:min-intrinsic;
               width:100px;
               background-image:url(theme:h-trackbar-back);
               background-repeat:stretch;
               vertical-align:middle;
        }

        :root > .slider
        {
             display:block;
           min-width:0.7em;
               foreground-image:url(theme:h-trackbar-thumb-normal);

               foreground-repeat:stretch;
        }
        :root:focus > .slider
        {
               foreground-image:url(theme:h-trackbar-thumb-focus);
        }

        :root > .slider:hover
        {
               foreground-image:url(theme:h-trackbar-thumb-hover);
        }
        :root > .slider:active
        {
               foreground-image:url(theme:h-trackbar-thumb-pressed);

        }

        :root:disabled > .slider
        {
               foreground-image:url(theme:h-trackbar-thumb-disable
d);
        }

        :root:tab-focus
        {
               outline:1px dotted invert -1px;
        }
      @media handheld
      {
           :root:tab-focus
           {
                outline:1px solid -2px highlight;
           }
      }
}

widget[type="vslider"],
input[type="vslider"]
{
        style-set: "std-vslider";
}

@set std-vslider
{
      :root
        {
               behavior:slider;
               cursor:default;
               padding:2px;
               height:100px;
               width:min-intrinsic;
               background-image:url(theme:v-trackbar-back);
               background-repeat:stretch;
        }
        :root:tab-focus
        {
               outline:1px dotted invert -1px;
        }
```

```
      :root > .slider
      {
        min-height:0.7em;
            foreground-image:url(theme:v-trackbar-thumb-normal);

            foreground-repeat:stretch;
      }

      :root:focus > .slider
      {
            foreground-image:url(theme:v-trackbar-thumb-focus);
      }

      :root > .slider:hover
      {
            foreground-image:url(theme:v-trackbar-thumb-hover);
      }
      :root  > .slider:active
      {
            foreground-image:url(theme:v-trackbar-thumb-pressed);

      }

      :root:disabled > .slider
      {
            foreground-image:url(theme:v-trackbar-thumb-disable
d);
      }

    @media handheld
    {
        :root:tab-focus
        {
            outline:1px solid -2px highlight;
        }
    }
}


widget[type="vscrollbar"],
input[type="vscrollbar"]
{
  behavior: scroll-bar;
  width:min-intrinsic;
  height:100%%;
}

widget[type="hscrollbar"],
input[type="hscrollbar"]
{
  behavior: scroll-bar;
  width:100%%;
  height:min-intrinsic;
}

frame
{
  behavior: frame;
  width:*;
  height:*;
}
iframe
{
  behavior: frame;
  width:300px;
  height:150px;
}

iframe[history],
frame[history]
{
  behavior: history frame;
}

frame[type="pager"]
{
  style-set: "std-pager";
}

@set std-pager
{
  :root
  {
    flow:null; // it will use manual layout
    behavior:pager;
    overflow:hidden;
  }

  :root > div.page
  {
    background-color:white;
    border: 1px solid black;
    outline: 3px glow black 1px;
    outline-shift: 2px;
}
}

frameset,
frameset[cols] { style-set: "std-frameset-cols"; }
frameset[rows] { style-set: "std-frameset-rows"; }

@set std-frameset-cols
{
  :root
  {
    behavior: frame-set;
    width:*;
    height:*;
    flow:horizontal;
  }
  :root[history] { behavior: history frame-set; }
  :root:not(:has-children-of-type(splitter)):not(:has-children-of-type
(hr))
  {
```

```
    border-spacing:3px;
    background-color:threedface;
  }
  :root > * { height:*; }
  :root > hr,
  :root > splitter { width:5px; cursor:e-resize; background-color: t
hreedface; margin:0; border:none; }
  :root > hr:hover,
  :root > splitter:hover { background-color:threedhighlight threed
shadow threedshadow threedhighlight; }
  /*:root > splitter:active { background:transparent url(theme:tool
bar-button-checked) stretch; }*/
}
@set std-frameset-rows
{
  :root
  {
    behavior: frame-set;
    width:*;
    height:*;
    flow:vertical;
  }
  :root[history] { behavior: history frame-set; }
  :root:not(:has-children-of-type(splitter)):not(:has-children-of-type
(hr))
  {
    border-spacing:3px;
    background-color:threedface;
  }
  :root > * { width:*; }
  :root > hr,
  :root > splitter { height:5px; cursor:n-resize; background-color:
threedface; margin:0; border:none; }
  :root > hr:hover,
  :root > splitter:hover { background-color:threedhighlight threed
highlight threedshadow threedshadow ; }
  /*:root > splitter:active { background:transparent url(theme:tool
bar-button-checked) stretch; }*/
}


@set std-popup-menu
  {
    :root, menu
    {
      behavior:menu; /*is a menu*/
      flow: vertical;
      display:none;
      margin:0;
      padding:1px;
      width:max-intrinsic;
      border:1px solid threedshadow;
      background-color: window;
      color:windowtext;
      margin:0 1px; /* to offset it from parent li */
      font: system-menu;
      cursor:default;
    }

    /* menu item in popup menus */
     menu > option,
    li
    {
      width:*;
      text-align:left;
      padding-left:24px; /* room for icon */
      padding-right:12px; /* room for the arrow */
      padding-top:4px;
      padding-bottom:4px;
      foreground-repeat: no-repeat;
      foreground-position: 2px 50%;
      color:windowtext;
    }
    menu > option:rtl,
    li:rtl
    {
      text-align:right;
      padding-right:24px; /* room for icon */
      padding-left:12px; /* room for the arrow */
      foreground-position-left:auto;
      foreground-position-right:2px;
    }

     //li[popup],
    //li:has-children
     menu > option:has-child-of-type(menu),
     li:has-child-of-type(menu)
    {
      background-image:url(stock:arrow-right); /* that arrow */
      background-repeat: no-repeat;
      background-position: 100% 50%;
    }

    menu > option:disabled,
    li:disabled
     {
        color:threedshadow; font-weight: normal;
        foreground-image-transformation: contrast-brightness-ga
mma( 0.15,0.75,1.0);
      }

    /* accesskey label (span) */
    span.accesskey
    {
      display:inline-block;
      margin-left:*; /* spring to attach it to the right */
      padding-left:1em;
      color:threeddarkshadow;
    }

    menu > option:current span.accesskey,
    li:current span.accesskey
    {
      color:brown;
```

```
    }

    /* current menu item */
    menu > option:current,
    li:current
    {
      background-color:highlight;
      color:highlighttext;
    }

    /* menu separator */
    hr
    {
      margin:2px;
    }
      caption
      {
        margin:2px;
        color:graytext;
      }
      img.hr
      {
        display:inline-block;
        vertical-align:bottom;
        width:*;
        height:0;
        margin-bottom:0.4em;
        border-top:1px solid threedshadow;
        border-bottom:1px solid threedlight;
      }
  }

menu.popup,
menu.context,
menu:synthetic  /* builtin popup menu */
{
  style-set: "std-popup-menu";
}

widget[type="calendar"],
input[type="calendar"]
{
  style-set: "std-calendar";
}

@set std-calendar /* style block for the content of the calendar
 */
  {
    :root
        {
      behavior:calendar;
      width:min-intrinsic;
      height:min-intrinsic;
      font:system;
      background:window;
      overflow:hidden;
      color:windowtext;
      padding:1px;
          border:1px solid threedshadow;
      white-space: nowrap;
      cursor:default;
        }

    :root:tab-focus /* :root here is the element with the style-se
t set */
    {
      outline:1px dotted invert -4px; /* inside */
    }

    table
    {
      width:*;
      height:*;
      border-spacing:0;
      padding:2px;
    }

    caption
    {
      flow:horizontal;
      line-height:1.8em;
      font:system;
      font-weight:bold;
    }

    div.prev-date
    {
      background-image:url(theme:h-scrollbar-minus-normal);
      background-repeat:stretch;
      width:2em;
      height:*;
      padding:2px;
      margin-right:*;
    }
    div.prev-date:rtl
    {
      margin-right:0;
      margin-left:*;
    }
    div.prev-date:active
    {
      background-image:url(theme:h-scrollbar-minus-pressed);
      background-repeat:stretch;
    }

    div.prev-date:hover {
      background-image:url(theme:h-scrollbar-minus-hover);
      background-repeat:stretch;
    }

    div.next-date
    {
      background-image:url(theme:h-scrollbar-plus-normal);
      background-repeat:stretch;
```

```
    width:2em;
    height:*;
    padding:2px;
    margin-left:*;
}
div.next-date:rtl
{
    margin-left:0;
    margin-right:*;
}
div.next-date:active
{
    background-image:url(theme:h-scrollbar-plus-pressed);
    background-repeat:stretch;
}
div.next-date:hover {
    background-image:url(theme:h-scrollbar-plus-hover);
    background-repeat:stretch;
}
td
{
    width: *;
    height: *;
    text-align:center;
    padding:1px 6px;
    border:1px solid transparent;
}

th
{
    font:8pt Tahoma;
    font-weight:normal;
    color:highlight;
    text-align:center;
    padding-top:4px;
    border-bottom:1px solid threedshadow;
}

td.month.off,
td.day.off
{
    color: brown;
}
td.day.other-month,
td.year.other-year,
td.decade.other-decade
{
    color: graytext;
}

td:current,
td:hover
{
    border:1px solid inactivecaption;
    border-radius: 2px;

    //outline:1px dotted highlight;
        background-color: threedhighlight threedhighlight inactivec
aption inactivecaption;
        //color:highlighttext;
    }


    :root:current td.month:current,
    :root:focus td.month:current,
    :root:current td.day:current,
    :root:focus td.day:current,
    :root:current td.year:current,
    :root:focus td.year:current,
    :root:current td.decade:current,
    :root:focus td.decade:current
    {
        background-color:threedface;
        border:1px solid highlight;
        border-radius: 2px;
        //background-color: threedhighlight threedhighlight inactiv
ecaption inactivecaption;
        //background-color:highlight;
        color:windowtext;
    }
    td:active {
        background-color:highlight;
        color:highlighttext;
    }
    td.today
    {
        border:1px solid red;
        border-radius:2px;
    }
    div.button
    {
        display:inline-block;
        padding:* 4px;
        text-align:center;
        height:*;
    }

    /* these are sensitive to the mouse wheel */
    div.button:hover
    {
        background-image:url(theme:toolbar-button-hover);
        background-repeat:expand;
    }
    div.button:active
    {
        background-image:url(theme:toolbar-button-pressed);
    }

    text.statusbar
    {
        border-top:1px solid threedshadow;
        padding:1px;
        line-height:1.4em;
    }
```

```
    span.today
    {
      display:inline-block;
      padding:2px;
    }

    span.today:hover {
        background-repeat:stretch;
        background-image:url(theme:toolbar-button-hover);
    }

    span.today:active {
        background-repeat:stretch;
        background-image:url(theme:toolbar-button-pressed);
    }

    span.today-legend
    {
      display:inline-block;
      border:1px solid red;
      border-radius:2px;
      width:15pt;
      height:10pt;
      margin:* * * 2px;
    }

  }

widget[type="date"],
input[type="date"]
{
    font:system;
    style-set: "std-date";
}

@set std-date < std-edit /* style block for the date ctl, derived
 from std-edit */
  {
    :root
    {
      behavior:date;
      cursor:default;
      padding:0;
      flow:horizontal;
      width:max-intrinsic;
      height:max-intrinsic;
      overflow:hidden;
      text-selection: highlighttext highlight;
    }
    :root > popup
    {
      background-color: window;
      padding:0;
      border: none;
      //transition:none;
      transition:window-slide-ttb;
    }
    :root > caption
    {
      height:*; width:*;
      min-width:max-intrinsic;
      margin:3px;
      behavior: masked-edit;
          cursor:default;
    }
    :root > button
    {
      display:block;
      transition:none;
      behavior: none;
      height:*;
      width: system-scrollbar-width;
      margin: system-border-width; /*1px 1px 1px 0px;*/
      padding:0;
      background-repeat:stretch;
      background-image:url(theme:combobox-button-normal);
    }
    :root > button:hover
    {
      transition:none;
      background-image:url(theme:combobox-button-hover);
    }
    :root > button:active
    {
      background-image:url(theme:combobox-button-pressed);
    }
    :root > button:disabled
    {
      background-image:url(theme:combobox-button-disabled);
    }
  }

input[type="masked"]
{
       style-set: "std-masked-edit";
}

@set std-masked-edit < std-edit /* style block for the masked c
tl, derived from std-edit */
  {
    :root[mask]
    {
      behavior:masked-edit;
      text-align:center;
      text-selection: highlighttext highlight;
    }
  }

widget[type="time"],
input[type="time"]
{
       style-set: "std-time-edit";
}
```

```
@set std-time-edit
  {
    :root
    {
      behavior:time;
      font:system;
      color:windowtext;
      height:1em;
      width:1em;
      background-image:url(theme:edit-normal);
      background-repeat:stretch;
      padding:system-border-width;
      min-width: min-intrinsic;
      min-height: min-intrinsic;
      cursor:default;
      overflow:hidden;
      white-space: nowrap;
          flow:grid;
          context-menu:url(res:behavior-edit-menu.htm);
          text-selection: highlighttext highlight;
    }

    :root > caption
    {
      left:1#;right:1#;
      top:1#;bottom:2#;
      height:*; width:*;
      min-height: min-intrinsic;
      padding:2px;
      behavior: masked-edit;
          cursor:default;
    }

    :root > caption:rtl
    {
      left:2#;right:2#;
    }

    :root:disabled
    {
      background-image:url(theme:edit-disabled);
          color:graytext;
      /*cursor:default;*/
    }

    :root[invalid]
    {
      color:red;
    }

    :root > button
    {
      behavior:clickable;
      cursor:default;
    }

    :root > button.minus
    {
      left:2#;right:2#;
      top:2#;bottom:2#;
      padding:0;
      margin: 0;
      background-image:url(theme:v-spin-minus-normal);
      background-repeat:stretch;
      width:system-scrollbar-width;
      height:50%%;
    }
    :root > button.minus:rtl
    {
      left:1#;right:1#;
    }

    :root > button.minus:hover
    {
          transition:none;
      background-image:url(theme:v-spin-minus-hover);
    }
    :root > button.minus:active
    {
      background-image:url(theme:v-spin-minus-pressed);
    }
    :root > button.minus:disabled
    {
      background-image:url(theme:v-spin-minus-disabled);
    }
    :root > button.plus
    {
      padding:0;
      left:2#;right:2#;
      top:1#;bottom:1#;
          margin: 0;
      background-image:url(theme:v-spin-plus-normal);
      background-repeat:stretch;
          width:system-scrollbar-width;
          height:50%%;
    }
    :root > button.plus:rtl
    {
      left:1#;right:1#;
    }

    @media handheld
        {
          :root
          {
              padding:3px;
              flow:horizontal;
          }
          :root > button.plus
          {
              height:100%%;
          }
          :root > button.minus
```

```
        {
            height:100%%;
        }
    }

    :root > button.plus:hover
    {
        transition:none;
      background-image:url(theme:v-spin-plus-hover);
    }
    :root > button.plus:active
    {
      background-image:url(theme:v-spin-plus-pressed);
    }
    :root > button.plus:disabled
    {
      background-image:url(theme:v-spin-plus-disabled);
    }
  }
```

九、模板语法

1、aardio 模板语法说明

不仅仅是做网页可以使用模板语法，用 HTMLayout 写桌面软件界面也可以使用相同的模板语法，
需要用到 HTML 文件的地方可以直接把后缀名改为 aardio，就可以在里面嵌入 aardio 代码了，页内的 HTML 超链接都可以改为 *.aardio 文件链接。

在 aardio 代码中直接支持 HTML 模板语法。
一、一个 aardio 文件，即可以是纯 aardio 代码，也可以是纯 HTML，也可以是 HTML、aardio 相互混合的模板代码，aardio 都能自动识别并解析。

二、aardio 源代码文件里可以直接书写 HTML 代码，在 HTML 代码中 aardio 代码必须置于 <? ..... ?> 内部，，类似 PHP 的模板语法，但是要注意与 PHP 的区别是，aardio 规定开始标记 <? 必须独立不能紧跟英文字母，例如 <?xml..... 不是合法的 aardio 代码段开始标记(aardio 会认为这是一个 XML 标记)。 另外，aardio 总是忽略?>后面或文件开始的空白字符（包含空格、制表符，换行，全角空格）。

三、在一个 aardio 文件中如果混合了 HTML 代码，aardio 将<? ..... ?> 之外的部分解析为： print("HTML 代码") 以调用全局函数 print 输出 HTML。
关于 print 函数的规则：
1)、print 允许接收多个参数，并且必须对每个参数调用 tostring()转换为字符串
2)、在一个 aardio 文件解析结束时，print 函数将收到一个 null 参数调用

四、可以使用 <?=表达式?> 输出文本，该代码的作用类似于 print(表达式)

五、aardio 文件应当并且只能以 UTF-8 编码保存，不建议添加 UTF8 BOM(如果添加了 BOM,aardio 仍然会自动移除)

六、aardio 编辑器支持识别以上语法编辑 HTML 模板文件，可直接将 HTML 模板代码复制到 aardio 编辑器中运行并生成网页。

aardio 开发环境也可以直接支持解析运行 HTML 模板，可新建一个 aardio 源码文全，复制下面的源代码粘帖到 aardio 编辑器中

```
<!doctype html>
<html><head><meta charset="utf-8"><title>帮助页面</title></head>
<body>当前时间<? = time() ?>
</body></html>
```

然后点【运行】按钮，可以看到立即生成了一个网页。

完整示例：

```
import win.ui;
/*DSG{{*/
var winform = win.form( text="HTMLayout 使用模板语法" )
/*}}*/

import web.layout;
var wbLayout = web.layout(winform)

wbLayout.html = /**
<!doctype html>
<html><head></head><body>
    <? for(i=1;100;1){ ?>
        <div>
            循环次数<?= i ?> 当前时间    <?= time() ?>
        </div>
    <? } ?>
</body>
</html>
**/

winform.show()
win.loopMessage();
```

2、HTMLayout 模板文件传递参数

所有 HTMLayout 中使用的 URL 链接，都可以直接在 URL 后面增加 URL 参数，例如：

```
<div id="header">
    <include src="titlebar.aardio?a=123&b=value" />
</div>
```

在 aardio 文件中使用 requset.get 获取 URL 参数（所有参数为字符串类型），例如：

```
<?=request.get["a"] ?>
```

或者用下面的方法获取参数也可以：

```
<?=(...).a ?>
```

wbLayout.go() 可以直接用第二个参数指定模板参数，例如：

```
wbLayout.go("/layout/main.aardio",{
        a="hello";
        b=123;
});
```

3、CSS 文件如何使用模板

实际上 HTMLayout 里任何文件都可以用 aardio 模板文件代替，CSS 当然也可以。

但是要注意你不能直接把一个 CSS 文件复制到 aardio 文件里，因为不符合 aardio 语法，CSS 文件需要在前面加上模板标记，例

如：

```
<?
        //这里是 aardio 代码，即使没有 aardio 代码也要有这个标记
?>
html,body{
    margin:0;
    height:100%;
    background:#fff;
    overflow:hidden;
}
```

也可以简单一点只写一个 ?>也行，例如：

```


?>
html,body{
    margin:0;
    height:100%;
    background:#fff;
    overflow:hidden;
}
```

这是为了告诉 aardio，后面的 CSS 代码不是 aardio 代码（ 不要再用 aardio 语法去执行了 ）
当然如果是 HTML 代码的话就不用这么做，aardio 文件默认就能自动识别出 HTML 语法。

3、图像文件如何使用模板（ 显示二维码的例子 ）

HTMLayout 里很酷的一个地方就是任何文件你都可以用模板来输出,
例如图像，下面是一个用模板显示二维码的例子：

首先在 aardio 工程中添加一个模板文件，例如 /images/qrcode.aardio，代码如下:

```
import qrencode.bitmap;
var qrBmp = qrencode.bitmap(request.get.data,tonumber(request.get.version),tonumber(request.get.level) );
print( qrBmp.copyBitmapString(tonumber(request.get.width) : qrBmp.width,"*.jpg",100) );//输出图像
```

然后在 HTMLayout 的页面中就可以直接显示二维码了，HTML 代码如下：

```
wbLayout.html = /**
<img src="\images\qrcode.aardio?data=www.aardio.com&level=2&width=300" style="margin:0 auto;">
**/
```

4、使用嵌入式服务器加载 aardio 模板

我们还可以使用 wsock.tcp.simpleHttpServer 启动一个嵌入式的 HTTP 服务器，
这时候就可以象在网站上编程一样可以直接使用 HTML 模板语法，例如：

```
import wsock.tcp.simpleHttpServer;
wbLayout.go( wsock.tcp.simpleHttpServer.startUrl("/html/main.aardio") );
```

非常方便的是 wsock.tcp.simpleHttpServer 可以自动分配可用端口，并且可以支持 aardio 资源目录下的嵌入文件，这个方法加载 aardio 模板可以适用于所有嵌入式浏览器组件，例如 web.form,web.kit,web.layout 等等。

十、常见问题

1、behavior 覆盖问题

这是最容易范的一个错误，例如在 CSS 中写 button{ behavior:my-behavior; } 然后就杯具了，button 的默认属性 behavior: button 被覆盖掉了，button 不再是 button，所以需要改为 button{ behavior:button my-behavior; } ，更好的方法是这样写 button{ behavior:~my-behavior; } 这里的波浪号表示节点已经拥有的 hehavior，所以~my-behavior 是追加而不是替换。

2、获取、修改 select 控件选中节点

在 HTMLayout 中 select 节点的下拉列表，在 HTML 中你可以使用 <option selected>选中节点</option> 指定选中的项目，不幸的是这个属性只在 HTML 中有效，运行时你需要使用节点的 checked 属性来代替，一个示例：

```
import win.ui;
/*DSG{{*/
winform = ..win.form(text="HTMLayout - Select 控件";right=599;bottom=399;)
/*}}*/

import web.layout;
wbLayout = web.layout(winform);

wbLayout.html =/***
<select #s>
	<option selected>测试 1</option>
	<option >测试 2</option>
</select>
***/
wbLayout.wait();

//获取当前选中节点
var ltEle = wbLayout.querySelector("#s option:checked")

//修改当前选中节点
wbLayout.querySelectorAll("#s option")[2].state.checked = true;

wbLayout.getEle("s").querySelector("option").parent().createElement("option","新增节点" );

winform.show()
win.loopMessage();
```

3、支持 label 标记

```
<input type="checkbox" #my-checkbox />
<label for="my-checkbox">Click me</label>
```

或者

```
<label>
	<input type="checkbox" />
```

```
    Click me
</label>
```

HTMLayout 里使用 # 简写 id 属性，会自动把 id 转化为小写，比如 #myCheckbox 相当于 id="mycheckbox" 而不是 id="myCheckbox"。

4、如何使 HTML 文本支持选区、复制等

HTMLayout 作为一个 UI 引擎，默认的网页中的文字都是不能选中的，当然也没法复制了，解决方法是在 HTML 中指定 behavior:htmlarea，示例:

```
import win.ui;
/*DSG{{*/
winform = win.form(text="HTMLayout - 文本选区( htmlarea )";right=599;bottom=399;)
/*}}*/

import web.layout;
wbLayout = web.layout(winform);

wbLayout.html =/***
<html>
  <head>
    <style>
      div.htmlarea
      {
        behavior:htmlarea;
        context-menu:selector(menu#for-htmlarea);
        cursor:text;
        border:1px dotted;
        padding:5px;
      }
    </style>
    <menu.context #for-htmlarea>
      <li command="htmlarea:copy" style="foreground-image:url(res:edit-copy.png)" >复制<span class="accesske
y">Ctrl+C</span></li>
      <li command="htmlarea:selectall" >全选<span class="accesskey">Ctrl+A</span></li>
    </menu>
  </head>
<body>
  <p>CSS 中的 behavior:htmlarea 属性可使节点支持文本选区:</p>
  <div .htmlarea tabindex=0>
<h3>这是一段测试文本</h3>
在这里点鼠标右键试试</br>
在这里按 CTRL + A 组合键试试
  </div>
</body>
</html>
***/
```

```
wbLayout.wait()

//全选文本
wbLayout.querySelector("div.htmlarea").xcall("selectAll")

//复制文本
wbLayout.querySelector("div.htmlarea").xcall("copy")

winform.show()
win.loopMessage();
```

5、HTML 中的 img 标记与 CSS 中的 foreground-image 属性

HTMLayout 中 img 节点的 src 属性会映射到 CSS 中的 foreground-image 属性,也就是等价于指定前景图像,img 显示图像的效果可以用 CSS 中的 foreground-**** 系列属性调整。例如 button 节点用 img 指定按钮图标，可添加下面的 CSS 使图标禁用时自动变灰：

```
button:disabled  img{
        foreground-image-transformation:colorize(graytext);
}
```

6、怎样动态改变背景、背景色

HTMLayout CSSS! 中并不支持 ele::background = "#FFF" 这种写法，CSSS! 脚本中修改背景必须对单个的背景属性分别赋值，例如：

```
self::background-color = rgb(34,34,136)
self::background-image = url(images/brick.png);
```

其中 self::background-color 只能使用 rgb 等函数创建颜色，不能指定数值，并且 rgb 的参数只能使用十进制数值。CSSS!虽然不支持，但是在 aardio 里修改背景颜色可以直接这样写

```
ltEle.style["background"] = "#FFF"
```

在 aardio 里修改节点的背景图片需要下面这样写：

```
ltEle.style["background-image"] = "url(" + ltEle.image +")";
```

也就是说，在 aardio 里修改 CSS 属性时使用的是 CSS 文件里的标准语法。

7、怎样在 CSSS!中获取、修改节点的文本

在 aardio 中获取或修改节点的文本 可以使用 ltEle.innerText 属性，而获取、修改控件的值使用 ltEle.value 属性就行了。

CSSS!里要特别注意的是对于 input 等控件，value 是一个状态而不是一个属性，示例：

```
input[type="text"]{
      active-on!:self:value = "这是文本框";
}
```

而对于 span,a 等非控件节点，value 状态值等价于 aardio 中的 ltEle.innerText。

一个完整的范例：

```
import win.ui;
```

```
/*DSG{{*/
winform = win.form(text="CSSS 中使用 value 状态值";right=599;bottom=399 )
/*}}*/

import web.layout;
wbLayout = web.layout(winform);

wbLayout.html =/***
<div id="my-div" > </div>
<input #my-button type="button" value="请点击这里">
***/

wbLayout.css = /**
#my-button{
	click!:$1(#my-div):value = "测试";
}
**/

winform.show()
win.loopMessage();
```

另外对于 span,a,p 等文本节点(但不支持 DIV 等布局用节点)，还可以在 CSS 中使用 content 属性修改内部文本，例如：

```
span#myid{
	content:"测试";
}
```

8、click 事件

HTMLayout 中一般指的 click 事件指的是按钮的点击事件，也就是 aardio 中的 onButtonClick 事件。
非按钮节点也可以显式的在 behavior 中指定 clickable 以支持该事件，也可以说 clickable 节点（按钮或显示指定 behavior:~clickable 的节点）才能触发 CSSS!中的 click 事件（也即 aardio 中 onButtonClick 事件）。

当你用鼠标点击一个非 clickable 节点时，在 aardio 中触发的不是 onButtonClick 事件，而是 onMouseClick 事件，非 clickable 节点在 CSSS!中触发的不是 click 事件而是 active-off 事件。

下面是一个简单范例，可点击上下两个节点比较其区别：

```
import win.ui;
/*DSG{{*/
winform = win.form(text="HTMLayout clickable 节点";right=599;bottom=399; )
/*}}*/

import web.layout;
wbLayout = web.layout(winform);

namespace web.layout.behavior.testBehavior{

	onButtonClick = function (ltTarget,ltEle,reason,behaviorParams) {
		ltEle.printf("onButtonClick 事件被触发")
```

```
    }

    onMouseClick = function (ltTarget,ltEle,x,y,ltMouseParams) {
        ltEle.printf("onMouseClick 事件被触发")
    }

    //所有不是 on 前缀的函数,都可以在 CSS 脚本中直接调用
    func = function(ltEle,a,b,c){
        ltEle.printf("调用了自定义函数，收到参数 a:%d b:%d c:%d ",a,b,c )
        return "返回新的值"
    }
}

wbLayout.html =/***
<div id="a">请点击这里 clickable 节点</div>
<div id="b">请点击这里 非 clickable 节点</div>
***/

wbLayout.css = /**
#a{
    behavior:testBehavior clickable;
}
#b{
    behavior:testBehavior;
}
**/

winform.show()
win.loopMessage();
```

9、behavior 中导入 aardio 名字空间

HTMLayout 中有一些内建的 behavior，不需要导入可直接使用，内建 behavior 以及官网文档：

accesskeys form hyperlink edit password currency url number decimal clickable button popup-menu radio check popup-selector progress select tree tree-checkmarks path-select file file-icon dropdown-select richtext plaintext slider scroll-bar frame history frame pager frame-set history frame-set menu calendar date masked-edit time

file-icon 这个 behavior 用来显示文件的图标，但是这个 behavior 只能用于 option 节点、并且只能显示 small,large 两个尺寸的图标，而标准库中的 shellIcon 与这个 behavior 用法类似，但支持更多的尺寸，icon-size 或 CSS 中的 -icon-size 属性支持 large,small,extralarge,syssmall,jumbo 等值，其中 jumbo 为 win7 超大图标(在 XP 上该值自动切换为 extralarge)

如果需要创建自定义的名字空间，则需要在 aardio 中的 web.layout.behavior 名字空间下创建子名字空间。下面是一个示例：

```
import win.ui;
/*DSG{{*/
winform = win.form(text="HTMLayout behavior 用法演示";right=599;bottom=399 )
/*}}*/
```

```
import web.layout;
var wbLayout = web.layout(winform);

namespace web.layout.behavior.myBehavior {
    onMouseClick = function (ltTarget,ltEle,x,y,ltMouseParams) {
        ltEle.printf("点击了节点，坐标 x:%d y:%d",x,y)
    }
}

wbLayout.html =/***
<div style="behavior:myBehavior;">请点击这里</div>
***/

winform.show()
win.loopMessage();
```

粗体字部分是关键代码，CSS 样式里的 behavior:myBehavior 指示 HTMLayout 导入 aardio 中的名字空间 web.layout.behavior.myBehavior 在该名字空间中可以接收、变更节点的交互行为。

你也可以在 aardio 工程里创建 web.layout.behavior.myBehavior 这个用户库。然后在 aardio 中使用 import web.layout.behavior.myBehavior 导入 behavior. 实际上在标准库中已经提供了很多可用的 behavior。

使用的时候用 import 语句在加载网页前导入就可以了。

另外要注意，如果你在 aardio 中定义了 web.layout.behavior.button.command 这样的名字空间（ 如果你的程序中有太多 behavior，用这样的名字空间归类可以更好的组织代码、改进可维护性 ）那么在 HTMLayout 中的 behavior 名字就是 button.command，但是你不可以在 CSS 中写 behavior:button.command，这是错的，正确的写法是 behavior:url(button.command)

10、使用默认浏览器打开 HTMLayout 页面超链接

```
import win.ui;
/*DSG{{*/
winform = win.form(text="使用默认浏览器打开 HTMLayout 页面超链接";right=599;bottom=399)
/*}}*/

import web.layout;
wbLayout = web.layout( winform );

wbLayout.html =/***
<a href="www.baidu.com">百度</div>
***/

import process;
wbLayout.sinking = {
    onHyperlinkClick = function (ltTarget,ltEle,reason,behaviorParams) {
        process.execute( ltTarget.href );
```

```
        return true;
    }
}

winform.show()
win.loopMessage();
```

11、theme url

theme url 用于获取系统主题相应的控件图片，格式为 "theme:控件名称-状态"，在 css 中也可以使用,例如

```
option > text:hover { background-image:url(theme:list-view-item-hover); }
```

示例:

效果图


```
import win.ui;
/*DSG{{*/
var winform = ..win.form( bottom=111;parent=...;text="Switch";right=263;max=false )
winform.add(  )
/*}}*/

import web.layout;

var wbLayout = web.layout(winform);

wbLayout.html = /**
<body>
<p>功能选择：</p>
<a type="switch"><em>杀毒软件</em></a>
<a type="switch"><em>防火墙</em></a>
</body>
**/

wbLayout.css = /**
html {
    font: system;
}
a[type="switch"] {
    padding-right: 20px;
    display: inline-block;
    behavior: check;
    background: url(theme:button-disabled) expand;
```

```
}
a[type="switch"] em {
    padding: 8px;
    display: inline-block;
    position: relative;
    font-style: normal;
    width: max-intrinsic;
    text-align: center;
    background: url(theme:button-normal) expand;
}
a[type="switch"]:hover em {
    background-image: url(theme:button-hover);
}
a[type="switch"]:active em {
    background-image: url(theme:button-pressed);
}
a[type="switch"]:checked em {
    left: 20px;
    transition: left(back-in-out, 200ms);
}
**/

winform.show();
win.loopMessage();
```

12、读写 DOM 节点属性应使用字符串类型

错误写法	正确写法
var layoutEle = wbLayout.getEle("my-button") layoutEle.selected = true ; layoutEle.selected = false ;	var layoutEle = wbLayout.getEle("my-button") layoutEle.selected = ""; layoutEle.selected = null;

13、使用 fixedrows 属性固定表格标题行

固定表格标题行这在浏览器中可能有些麻烦，但在 HTMLayout 中非常简单。table 节点(表格) 可使用 fixedrows 属性指定在滚动时固定位置的标题行数。

例如 <table fixedrows ="2"> 表示第一行，第二行在滚动表格时固定位置 。

需要同时在 CSS 中指定表格的高度，并指定 CSS 属性 overflow: auto; 以启用滚动条。

效果图

table节点(表格) 可使用fixedrows属性指定在滚动时固定位置的标题行数。

Character	Entity	Description
Â	Â	capital A with circumflex
Ã	Ã	capital A with tilde
Ä	Ä	capital A with diaeresis
Å	Å	capital A with ring above = capital A ring
Æ	Æ	capital AE = capital ligature AE
Ç	Ç	capital C with cedilla
è	è	capital E with grave
é	é	capital E with acute
ê	ê	capital E with circumflex
Ë	Ë	capital E with diaeresis

示例如下(粗体字部分为关键代码):

```
import win.ui;
/*DSG{{*/
var winform = ..win.form( bottom=327;parent=...;right=593;text="使用 fixedrows 属性固定表格标题行" )
winform.add(  )
/*}}*/

import web.layout;
wbLayout = web.layout(winform);

wbLayout.html = /**
<body>
table 节点(表格) 可使用 fixedrows 属性指定在滚动时固定位置的标题行数。
<table fixedrows="1" cellspacing="0" >
  <tr><th>Character<th>Entity<th>Description</th></tr>
  <tr><td>à<td>à<td><font color="brown">capital</font> A <font color="green">with</font> grave = <font color="brown">capital</font> A grave</td></tr>
  <tr><td>á<td>á<td><font color="brown">capital</font> A with acute</td></tr>
  <tr><td>Â<td>Â<td><font color="brown">capital</font> A with circumflex</td></tr>
  <tr><td>Ã<td>Ã<td><font color="brown">capital</font> A with tilde</td></tr>
  <tr><td>Ä<td>Ä<td><font color="brown">capital</font> A with diaeresis</td></tr>
  <tr><td>Å<td>Å<td><font color="brown">capital</font> A with ring above = <font color="brown">capital</font> A ring</td></tr>
  <tr><td>Æ<td>Æ<td><font color="brown">capital</font> AE = <font color="brown">capital</font> ligature AE</td></tr>
  <tr><td>Ç<td>Ç<td><font color="brown">capital</font> C with cedilla</td></tr>
  <tr><td>è<td>è<td><font color="brown">capital</font> E with grave</td></tr>
  <tr><td>é<td>é<td><font color="brown">capital</font> E with acute</td></tr>
  <tr><td>ê<td>ê<td><font color="brown">capital</font> E with circumflex</td></tr>
  <tr><td>Ë<td>Ë<td><font color="brown">capital</font> E with diaeresis</td></tr>
</table>
</body>
**/

wbLayout.css = /**
body{
  font:system;
}
table{
  margin:20px;
  height:200px;
  overflow: auto;
  width:100%;
}
table th {
  color: white;
```

```
  font-family:"Century Gothic","Verdana";
  font-size:10pt;
  border:none;
  padding:4px;
  background-color:#666;
}
**/
winform.show()
win.loopMessage();
```

14、_service 节点

在 CSS 中_service 用于表示 HTMLayout 引擎内部合成的节点
示例:

```
import win.ui;
/*DSG{{*/
var winform = ..win.form( bottom=399;parent=...;text="_service 标签";right=599 )
winform.add(  )
/*}}*/

import web.layout;
wbLayout = web.layout(winform)
import web.layout.debug;
wbLayout.attachEventHandler( web.layout.debug );
wbLayout.html = /**
<body>
  <a title="测试提示文字">鼠标悬停在这里试试</a>
</body>
**/

wbLayout.css = /**
_service popup{
  background:yellow;
  width:max-intrinsic;
  height:max-intrinsic;
  border-radius:4px;
  padding:20px;
}

body{
  margin:20px;
  font:system 10.5pt;
}
**/

winform.show()
win.loopMessage();
```

_service popup 在这里表示鼠标悬停在节点时，HTMLayout 根据 title 属性自动创建的提示框

15、stock url

HTMLayout 支持一些自定义的协议，例如 data:.... res:.... stock:.....，格式都类似。
stock 实际上仅用于绘制方向箭头（实际上是内置的几个 png 文件），可用于替代图像 URL，下面是使用范例:

```
import win.ui;
/*DSG{{*/
var winform = ..win.form( bottom=399;parent=...;text="aardio Form";right=599 )
winform.add(  )
/*}}*/

import web.layout;
wbLayout = web.layout( winform )

wbLayout.html = /**
  <img src="stock:arrow-left" ></img>
  <img src="stock:arrow-right" ></img>
  <img src="stock:arrow-up" ></img>
  <img src="stock:arrow-down" ></img>
  <hr />
  <div #left >  </div>
**/

wbLayout.css = /**
#left {
    background:url(stock:arrow-left) no-repeat 0 0 ;
    width:48px;
    height:48px;
}
**/

winform.show()
win.loopMessage();
```

16、data url

data url 用于直接内嵌资源文件，可以直接用于 html 中，例如 img 图像节点的 src 属性。下面是使用范例:

```
import win.ui;
/*DSG{{*/
var winform = ..win.form( bottom=399;parent=...;right=599;text="aardio Form" )
winform.add(  )
/*}}*/

import web.layout;
var wbLayout = web.layout(winform);
```

```
wbLayout.html = /*********
<html>
  <head>
    <style>
      div.test  {
        background-image: url(
          R0lGODdhMAAwAPAAAAAAAP///ywAAAAAMAAw
          AAAC8IyPqcvt3wCcDkiLc7C0qwyGHhSWpjQu5yqmCYsapyuvUUlvONmOZtfzgFz
          ByTB10QgxOR0TqBQejhRNzOfkVJ+5YiUqrXF5Y5lKh/DeuNcP5yLWGsEbtLiOSp
          a/TPg7JpJHxyendzWTBfX0cxOnKPjgBzi4diinWGdkF8kjdfnycQZXZeYGejmJl
          ZeGl9i2icVqaNVailT6F5iJ90m6mvuTS4OK05M0vDk0Q4XUtwvKOzrcd3iq9uis
          F81M1OIcR7lEewwcLp7tuNNkM3uNna3F2JQFo97Vriy/Xl4/f1cf5VWzXyym7PH
          hhx4dbgYKAAA7);
        background-repeat:repeat;
        width:200px;
        height:200px;
        border:1px solid black;
      }

    </style>
  </head>
<body>
  <h1> Data URL 演示</h1>
   <div .test />
</body>
</html>
*********/

winform.show()
win.loopMessage();
```

十一、API 文档

1、web.layout

web.layout 成员列表

web.layout._dll

```
dll;
```

web.layout.appendMasterCss(css)

追加默认 CSS，必须在创建 HTMLayout 窗体,以及调用其他函数以前调用

web.layout.appendMasterCssFile("/res/master->css")

追加默认 CSS 文件
CSS 文件请使用 UTF 编码保存，必须在创建 HTMLayout 窗体,以及调用其他函数以前调用

web.layout.createEle()
返回对象:layoutEleObject

web.layout.createEle(标签名,节点内容)
创建节点，节点内容可省略

web.layout.declareEle("__", _HLELM___)
声明标记，必须在创建 HTMLayout 窗体,以及调用其他函数以前调用，并且须要早于设置 MasterCss 的函数调用

web.layout.setMasterCss(css)
替换默认 CSS，必须在调用其他函数之前调用

全局对象 成员列表

_event_subsriptions
在 HTMLayout 中的 behavior 对象,或 eventHandle 对象中可添加此成员变量，使用一个或多个 HL_HANDLE 前缀的掩码来指定指定订阅的事件，如果未显示指定该参数,程序将自动设定为合适的值

web 成员列表

web.layout(窗口对象,捕获事件)
从窗口对象创建 HTMLayout 窗口，参数二默认为 _HL_HANDLE_LITE - 仅允许捕获按钮命令等基本交互事件

weblayoutObject 成员列表

weblayoutObject.$("标签名[属性名='属性值']")
使用 CSS 选择器语法，查找所有符合条件节点

weblayoutObject.$()
返回对象:layoutEleObject

weblayoutObject.$1("标签名[属性名='属性值']")
使用 CSS 选择器语法，查找第一个符合条件节点

weblayoutObject.$1()
返回对象:layoutEleObject

weblayoutObject._contrls[]
返回对象:staticObject

weblayoutObject.addHeaders

```
weblayoutObject.addHeaders = /**
```

```
Accept-Language: cn
/*设置默认的 HTTP 请求头，多个请求头使用回车换行分隔*/
**/
```

weblayoutObject.attachEventHandler(eventHandler,subscription)

添加事件监听对象,参数一不可省略，subscription 可选使用 HL_HANDLE 前缀常量指定捕获的事件，省略该参数则根据定义的回调函数自动设定该值，也可以使用对象的 _event_subsriptions 成员指定该值，该函数返回事件 ID,用于注销监听

weblayoutObject.callback

```
weblayoutObject.callback = function ( message,wParam,lParam,vParam,notifyCode ) {

}
```

weblayoutObject.combineUrl(URL)

将 URL 转换为绝对路径，可省略参数返回 baseURL

weblayoutObject.commitUpdates()

立即更新所有节点

因为节点可能会延迟更新，需要立即获取节点更新后的位置可调用此函数，该函数不会调用 updateWindow

weblayoutObject.createEle()

返回对象:layoutEleObject

weblayoutObject.createEle(标签名,节点内容)

创建节点，节点内容可省略

weblayoutObject.css

```
weblayoutObject.css = /**
#my-button{
    behavior:"my.command";
    active-on!:
        /*输入 CSS 脚本,语句用逗号分隔*/
    ;
}
**/
```

weblayoutObject.debug()

```
import web.layout.debug;
weblayoutObject.attachEventHandler( web.layout.debug );
```

weblayoutObject.detachEventHandler(事件 ID)

注销事件监听对象，事件 ID 不可省略

weblayoutObject.documentElement

返回对象:layoutEleObject

weblayoutObject.enumQuery(枚举函数,CSS 选择器,格式化参数)

```
weblayoutObject.enumQuery(
    function(ltEle){
        /*返回 true 停止枚举*/
    },"div"
)
```

weblayoutObject.enumRes(枚举函数)

```
weblayoutObject.enumRes(
    function(uri,resType,imgData,size){
        /*枚举页面页源*/
    }
)
```

weblayoutObject.eventsHandler

```
weblayoutObject.eventsHandler = function (tag,he,evtg,prms) {

}
```

weblayoutObject.fromPoint()

返回对象:layoutEleObject

weblayoutObject.fromPoint(x 坐标,y 坐标,是否屏幕坐标)

参数三可选,默认为窗口客户区坐标，成功返回节点对象

weblayoutObject.getCtrl("")

根据 ID 或 name 查找节点，并获取在该节点中自定义的控件对象

weblayoutObject.getCtrl()

返回对象:staticObject

weblayoutObject.getEle("__")

根据 ID 查找节点，如果未找到,尝试使用 name 查找

weblayoutObject.getEle()

返回对象:layoutEleObject

weblayoutObject.getEleByUid()

返回对象:layoutEleObject

weblayoutObject.getEleByUid(UID)

根据唯一标志符(数值)返回节点

weblayoutObject.getEles("__")

根据 name 属性查找节点

weblayoutObject.getEles()

返回对象:layoutEleObject

weblayoutObject.getFocus()

获取当前输入焦点所在节点

返回对象:layoutEleObject

weblayoutObject.go('__/*路径*/')

跳转到指定网址(支持资源文件)

weblayoutObject.html

```
weblayoutObject.html = /**
<!doctype html>
<html>
<head>
    <style type="text/css">
    html,body{ height:100%; margin:0; }
    </style>
</head>
<body>
    <div id="header"></div>
    <div id="container">
        <div class="lside"> </div>
        <div class="rside"> </div>
    </div>
</body>
</html>
**/
```

weblayoutObject.location

当前页面 URL，只读属性

weblayoutObject.onControlCreated

```
weblayoutObject.onControlCreated = {
    节点 ID = function( ltEle,ltCtrl ){
        /*onControlCreate 也可以是一个函数,在 onElementControlCreated 之前被触发*/
    }
}
```

weblayoutObject.onDocumentComplete

```
weblayoutObject.onDocumentComplete = function() {
    /*网页加载完成*/;
}
```

weblayoutObject.post(网址,参数)

参数可以是字符串或键值对组成的表对象，该函数可触发 onDataArrived 事件

weblayoutObject.queryEle()

返回对象:layoutEleObject

weblayoutObject.queryEles()

返回对象:layoutEleObject

weblayoutObject.querySelector("标签名[属性名='属性值']")

使用 CSS 选择器语法，查找第一个符合条件节点，如果有多个参数则首先调用 string.format 格式化为 CSS 文本，该函数名可使用'$1'代替,等价于 CSS!中的$1 函数

weblayoutObject.querySelector()

返回对象:layoutEleObject

weblayoutObject.querySelectorAll("标签名[属性名='属性值']")

使用 CSS 选择器语法，查找所有符合条件节点，如果有多个参数则首先调用 string.format 格式化为 CSS 文本，该函数名可使用'$'代替,等价于 CSS!中的$函数

weblayoutObject.querySelectorAll()

返回对象:layoutEleObject

weblayoutObject.readyState

获取当前状态,可能返回的值：'uninitialized','loading','complete'

weblayoutObject.setCss(css,mediaType,baseUrl)

替换 CSS

weblayoutObject.setMode(_HLM__)

设置模式

_HLM_SHOW_SELECTION 类似于在 body 节点应用 behavior:htmlarea

weblayoutObject.setOption(_HL_OPTIONS__,1)

设置选项

weblayoutObject.traverseUiEvent(结构体参数)

结构体参数是 web.layout.event.MOUSE_PARAMS 或 KEY_PARAMS

该函数发送事件经过捕获与冒泡过程直到事件被处理

weblayoutObject.updateWindow()

更新窗口，如果仅仅需要更新节点位置等,可调用 commitUpdates 函数

weblayoutObject.wait()

等待所有资源加载完成

weblayoutObject.write(__/*HTML 代码*/)

写出指定 HTML 代码

全局常量函数

::HTMLayout=HTMLayout.dll
返回对象:dllModuleObject

自动完成常量

BLOCK_BLOCK_ELEMENT=4
_BLOCK_TEXT_ELEMENT=3
_DATA_ELEMENT=0
_HLELM_BLOCK_BLOCK_ELEMENT=4
HLELM_BLOCK_TEXT_ELEMENT=3
HLELM_DATA_ELEMENT=0
HLELM_INLINE_BLOCK_ELEMENT=2
HLELM_INLINE_TEXT_ELEMENT=1
_HLM_LAYOUT_ONLY=0
_HLM_SHOW_SELECTION=1
_HLN_ATTACH_BEHAVIOR=0xB07
_HLN_CONTROL_CREATED=0xB02
_HLN_CREATE_CONTROL=0xB00
_HLN_DATA_LOADED=0xB03
_HLN_DESTROY_CONTROL=0xB06
_HLN_DIALOG_CLOSE_RQ=0xB10
_HLN_DIALOG_CREATED=0xB0F
_HLN_DOCUMENT_COMPLETE=0xB04
_HLN_LOAD_DATA=0xB01
_HLN_UPDATE_UI=0xB05
_HLRT_DATA_CURSOR=3
_HLRT_DATA_HTML=0
_HLRT_DATA_IMAGE=1
_HLRT_DATA_SCRIPT=4
_HLRT_DATA_STYLE=2
_HL_ANIMATION_THREAD=5
_HL_CONNECTION_TIMEOUT=2
_HL_DISABLE_INITIALIZATION=0x80000000
_HL_FONT_SMOOTHING=4
_HL_HANDLE_ALL=0xFFFF
_HL_HANDLE_BEHAVIOR_EVENT=0x100
_HL_HANDLE_DATA_ARRIVED=0x80
_HL_HANDLE_DRAW=0x40
_HL_HANDLE_EXCHANGE=0x1000
_HL_HANDLE_FOCUS=4
_HL_HANDLE_GESTURE=0x2000
_HL_HANDLE_INITIALIZATION=0
_HL_HANDLE_INPUT=0x80000007
_HL_HANDLE_KEY=2
_HL_HANDLE_LITE=0x80000100
_HL_HANDLE_METHOD_CALL=0x200
_HL_HANDLE_MOUSE=1
_HL_HANDLE_SCROLL=8
_HL_HANDLE_SIZE=0x20
_HL_HANDLE_TIMER=0x10
_HL_HTTPS_ERROR=3
_HL_OPTIONS_ANIMATION_THREAD=5
_HL_OPTIONS_CONNECTION_TIMEOUT=2
_HL_OPTIONS_FONT_SMOOTHING=4
_HL_OPTIONS_HTTPS_ERROR=3
_HL_OPTIONS_SMOOTH_SCROLL=1
_HL_OPTIONS_TRANSPARENT_WINDOW=6
_HL_SMOOTH_SCROLL=1
_HL_TRANSPARENT_WINDOW=6
_HWND_DISCARD_CREATION=1
_HWND_TRY_DEFAULT=0
_INLINE_BLOCK_ELEMENT=2
_INLINE_TEXT_ELEMENT=1
_LOAD_DISCARD=1
_LOAD_OK=0
_SIH_APPEND_AFTER_LAST=2
_SIH_INSERT_AT_START=1
_SIH_REPLACE_CONTENT=0
_SOH_INSERT_AFTER=5
_SOH_INSERT_BEFORE=4
_SOH_REPLACE=3

2、web.layout.behavior

web.layout.behavior 成员列表

web.layout.behavior.__release(this)

使用类声明的 behavior 对象，必须在 onDetach 事件中使用此函数注销自身，如果未定义 onDetach 事件将会自动添加该事件

web.layout.behavior.collapsibleByIcon

折叠按钮(图标)

web.layout.behavior.collapsibleList

折叠列表(项目可展开单项或全部折叠)

web.layout.behavior.dragable

使节点可拖动

样式中可使用 -drag-margin:"上 右 下 左"指定拖动边距属性以限制拖动范围，不限制拖动范围则可以将节点拖出窗口，并触发onPopup***系列事件，节点可使用 dragable 属性设置是否允许在该节点开始拖动

web.layout.behavior.dropdown

下拉选框

web.layout.behavior.expandableList

扩展列表(实现手风琴效果/Outlook 风格菜单)

web.layout.behavior.grid

网格控件

web.layout.behavior.gripper

用于创建拖动手柄，拖动该对象时拖动父节点

web.layout.behavior.lightBoxDialog

高亮对话框效果

使用 import 语句在加载页面前导入即可生效

web.layout.behavior.path

文件路径显示支持

超出显示宽度显示省略号,鼠标悬停提示完整内容

web.layout.behavior.popup

弹出面板

web.layout.behavior.scroller

自动滚动效果

web.layout.behavior.selectionBox

鼠标选框

web.layout.behavior.shellIcon

文件图标效果

使用 import 语句在加载页面前导入即可生效

web.layout.behavior.sizer

使对象可通过鼠标拖动调整大小

web.layout.behavior.sortableGrid

排序网格控件

web.layout.behavior.splitter

拆分条

web.layout.behavior.tabs

HTMLayout tabs behavior

使用 import 语句在加载页面前导入即可生效

web.layout.behavior.windowCommand

支持窗口命令,如最大化,最小化,关闭等等

web.layout.behavior.windowSizer

用于无边框窗口以支持窗体大小可调边框

3、web.layout.element

全局对象 成员列表

layoutEle2

返回对象:layoutEleObject

ltEle

HTMLayout 节点对象，lt 前缀变量为 HTMLayout 保留，请勿用于表示其他类型对象

返回对象:layoutEleObject

ltEle2

返回对象:layoutEleObject

ltOption

HTMLayout 节点对象，lt 前缀变量为 HTMLayout 保留，请勿用于表示其他类型对象

返回对象:layoutEleObject

ltPopupOwner

HTMLayout 节点对象，lt 前缀变量为 HTMLayout 保留，请勿用于表示其他类型对象

返回对象:layoutEleObject

ltTarget

HTMLayout 节点对象，在事件函数中，ltTarget 一般表示触发事件的目标节点，lt 前缀变量为 HTMLayout 保留，请勿用于表示其他类型对象

返回对象:layoutEleObject

layoutEle 成员列表

HTMLayout 节点对象保留变量名，请勿用于表示其他类型对象
返回对象:layoutEleObject

layoutEleObject 成员列表

layoutEleObject.$("标签名[属性名='属性值']")

使用 CSS 选择器语法，查找符合条件的所有子节点

layoutEleObject.$()

返回对象:layoutEleObject

layoutEleObject.$1("标签名[属性名='属性值']")

使用 CSS 选择器语法，查找符合条件的第一个子节点

layoutEleObject.$1()

返回对象:layoutEleObject

layoutEleObject.$1p("标签名[属性名='属性值']")

使用 CSS 选择器语法，查找符合条件最近的父节点，注意与 CSS 脚本不同，该函数测试自身

layoutEleObject.$1p()

返回对象:layoutEleObject

layoutEleObject.$p("标签名[属性名='属性值']")

使用 CSS 选择器语法，查找符合条件的所有父节点，注意与 CSS 脚本不同，该函数测试自身

layoutEleObject.$p()

返回对象:layoutEleObject

layoutEleObject.animate(动画回调函数，延时毫秒)

```
layoutEleObject.animate(
    function(ltEle，step){
        if( step == 0xffffffff )return 0;

        return /*新的延时值，返回 0 停止*/;
    }
)
```

layoutEleObject.append(__/*节点*/)

追加到最后一个子节点

layoutEleObject.attachEventHandler(eventHandler，subscription)

添加事件监听对象，参数一指定响应事件的对象，如果不指定则为当前节点对象，subscription 可选使用 HL_HANDLE 前缀常量指定捕获的事件，如果不指定该参数，则根据预听义的监听函数自动设定该值，也可以使用对象的 _event_subsriptions 成员指定该值，该函数返回事件 ID，用于注销监听.

layoutEleObject.capture

设为 true 开始捕获鼠标消息，设为 false 取消捕获

layoutEleObject.child()

返回对象:layoutEleObject

layoutEleObject.child(1__/*索引*/)

获取子节点

第一个子节点的索引为 1

layoutEleObject.childCount()

获取子节点数目

返回对象:layoutEleObject

layoutEleObject.clearAttributes()

清空所有属性

layoutEleObject.click()

触发 onButtonClick 事件，支持按钮节点、或者样式中指定 behavior:~clickable 的节点

触发非 clickable 的 onMouseClick 事件请使用 traverseMouse 函数

layoutEleObject.clone()

返回对象:layoutEleObject

layoutEleObject.clone(索引)

复制节点，必须添加为其他节点的子节点才能使用。

可选指定一个位置参数以添加到复制源节点的父节点，-1 表示追加到兄弟节点尾部

返回对象:layoutEleObject

layoutEleObject.cls

自定义控件类名

layoutEleObject.combineUrl(URL)

将相对路径转换为绝对路径

layoutEleObject.createElement()

返回对象:layoutEleObject

layoutEleObject.createElement(标签名，节点内容，插入位置)

创建节点，节点内容可省略，插入位置可省回，默认为-1 表示追到子节点尾部

layoutEleObject.createKeyEvent()

返回对象:hlKeyParamsObject

layoutEleObject.createKeyEvent(虚拟键码，ltTarget)

创建键盘事件，参数 2 可选

layoutEleObject.createMouseEvent()

返回对象:hlMouseParamsObject

layoutEleObject.createMouseEvent(x，y，ltTarget)

创建鼠标事件，所有参数可选

layoutEleObject.delayMeasure()

延迟更新布局，用于拖动时优化性能

layoutEleObject.delete()

删除节点对象

成功返回 true

layoutEleObject.detach()

自 DOM 树中分离该节点

当引用计数清零时自动删除

layoutEleObject.detachEventHandler(事件 ID)

注销事件监听对象，省略参数则停止节点自身事件监听

layoutEleObject.documentElement

当前当前文档 HTML 根节点

返回对象:layoutEleObject

layoutEleObject.eachAttribute()

```
for( i，name，value in layoutEleObject.eachAttribute() ){

}
```

layoutEleObject.eachChild()

返回对象:layoutEleObject

layoutEleObject.eachChild(开始索引，结束索引，步进)

```
for(i，eleChild in layoutEleObject.eachChild(/*可选指定开始与结束索引*/) ){
    io.print( i，eleChild.innerHTML );
}
```

layoutEleObject.enumQuery(枚举函数，CSS 选择器，格式化参数)

```
layoutEleObject.enumQuery(
    function(ltEle){
        /*返回 true 停止枚举*/
    }，"div"
)
```

layoutEleObject.enumStyle(枚举函数)

```
layoutEleObject.enumStyle(
```

```
    function(selector，ruleType，file，lineNo){

    }
)
```

layoutEleObject.firstSibling()

获取第一个兄弟节点

返回对象:layoutEleObject

layoutEleObject.getAttribute("hide")

获取属性，返回值为字符串或 null

也可以直接写 var 返回值=ele.属性名字

layoutEleObject.getCaretPos()

返回光标相对于当前节点的客户区位置，返回值分别为:左坐标，顶坐标，宽度，高度

layoutEleObject.getCharFromPos(x，y)

返回文本控件指定坐标文本索引

如果存在子节点，返回文本索引，子节点对象，子节点文本索引

layoutEleObject.getCtrl()

返回自定义控件对象

返回对象:staticObject

layoutEleObject.getCustomAttribute("__")

获取 HTML 属性，如果失败，则在名字前添加横线查找 style 自定义属性

layoutEleObject.getDataTable()

返回节点 data-table 属性中的值，并序列化为 table 对象

layoutEleObject.getElementById("__")

根据 ID 查找节点

如果未找到，尝试使用 name 查找

layoutEleObject.getElementById()

返回对象:layoutEleObject

layoutEleObject.getElementByUid()

返回对象:layoutEleObject

layoutEleObject.getElementByUid(UID)

根据唯一 ID 在该节点所属窗口范围内查找节点

layoutEleObject.getElementsByName ("__")

根据 name 查找节点集合

layoutEleObject.getElementsByName()

返回对象:layoutEleObject

layoutEleObject.getElementsByTagName("__")

根据 HTML 标签名查找节点集合

layoutEleObject.getElementsByTagName()

返回对象:layoutEleObject

layoutEleObject.getForm()

返回该节点所在窗体对象

返回对象:winform

layoutEleObject.getForm(false)

返回节点所在窗口或控件对象

返回对象:staticObject

layoutEleObject.getHwnd(是否返回顶层窗口)

返回窗口句柄，参数可省略，默认为 false

layoutEleObject.getIntrinsicHeight()

返回高度的 min-intrinsic 值

layoutEleObject.getIntrinsicWidth()

返回宽度的 min-intrinsic，max-intrinsic 值

layoutEleObject.getLayout()

返回 HTMLayout 窗体对象

返回对象:weblayoutObject

layoutEleObject.getPos()

返回文档相对 x 坐标，y 坐标，cx 宽度，cy 高度

layoutEleObject.getRect()

返回对象:rectObject

layoutEleObject.getRect(_HL__)

返回节点区块坐标

参数可省略，默认值为 _HL_ROOT_RELATIVE | _HL_CONTENT_BOX

layoutEleObject.getScrollInfo()

获取滚动条信息，

即使 CSS 样式中使用 overflow:hidden;隐藏滚动条，仍然可以正常取到滚动分页等参数，

返回对象:ltEleScrollInfoObject

layoutEleObject.getState(htmlayout.ELEMENT_STATE_BITS.__)

是否有某个状态，以 HL_STATE 开头的常量表示指定状态

layoutEleObject.getValueObject()

返回控件值对象(web.layout.valueObject 对象)

返回对象:webLayoutValueObject

layoutEleObject.getsel()

返回文本框控件文本选区，返回值:起始位置，结束位置

layoutEleObject.id

节点 ID

layoutEleObject.index()

节点在父节点 child 子节点集合中的索引位置

layoutEleObject.innerHTML

节点内部 HTML 源码

layoutEleObject.innerText

设置或者获取文本

layoutEleObject.insert(节点，插入位置)

插入子节点，参数 2 可省略，默认插入子节点队列最前面

插入位置为-1 表示插入到子节点队列尾部

layoutEleObject.insertAdjacentHTML("afterBegin"，__/*HTML 代码*/)

在节点内部最前面插入 HTML

可添加多个 HTML 参数

layoutEleObject.insertAdjacentHTML("afterEnd"，__/*HTML 代码*/)

在节点之后插入 HTML

可添加多个 HTML 参数

layoutEleObject.insertAdjacentHTML("beforeBegin"，__/*HTML 代码*/)

在节点之前插入 HTML

可添加多个 HTML 参数,

layoutEleObject.insertAdjacentHTML("beforeEnd"，__/*HTML 代码*/)

在节点内部最后面插入 HTML

可添加多个 HTML 参数

layoutEleObject.isChild(子节点)

判断参数中的节点是否子节点或自身

参数为空值返回空

layoutEleObject.isEnabled()

节点是否可用

layoutEleObject.isParent(父节点)

判断参数中的节点是否父节点或自身
参数为空值返回空

layoutEleObject.isVisible()

节点是否可见

layoutEleObject.lastSibling()

获取最后一个兄弟节点
返回对象:layoutEleObject

layoutEleObject.modifyState(要增加的状态，要取消的状态，是否更新)

设置状态，以 HL_STATE 开头的常量表示指定状态
所有参数可省略，参数三默认为 true

layoutEleObject.name

节点 name 属性

layoutEleObject.next()

下一个兄弟节点
返回对象:layoutEleObject

layoutEleObject.nextSibling()

获取下一个兄弟节点，可选在参数中指定偏移值
返回对象:layoutEleObject

layoutEleObject.onActiveChild

```
layoutEleObject.onActiveChild = function (ltTarget，ltOwner，reason，behaviorParams) {

}
```

layoutEleObject.onAnimation

```
layoutEleObject.onAnimation = function (ltTarget，ltOwner，reason，behaviorParams) {

}
```

layoutEleObject.onApplicationEvent

```
layoutEleObject.onApplicationEvent = function (ltTarget，ltOwner，reason，behaviorParams) {
    /*postEvent 或 sendEvent 发出一大于 0x100 的消息*/
}
```

layoutEleObject.onButtonClick

```
layoutEleObject.onButtonClick = function (ltTarget，ltOwner，reason，behaviorParams) {

}
```

layoutEleObject.onButtonPress

```
layoutEleObject.onButtonPress = function (ltTarget, ltOwner, reason, behaviorParams) {

}
```

layoutEleObject.onButtonStateChanged

```
layoutEleObject.onButtonStateChanged = function (ltTarget, ltOwner, reason, behaviorParams) {

}
```

layoutEleObject.onClick

```
layoutEleObject.onClick = function( ltOwner ){
    /*click 函数触发此事件*/
    return true;
}
```

layoutEleObject.onClosePopup

```
layoutEleObject.onClosePopup = function (ltTarget, ltOwner, reason, behaviorParams) {

}
```

layoutEleObject.onContextMenuRequest

```
layoutEleObject.onContextMenuRequest = function (ltTarget, ltOwner, reason, behaviorParams){
    var ltPopupOwner = ..web.layout.element( behaviorParams.he )

}
```

layoutEleObject.onContextMenuSetup

```
layoutEleObject.onContextMenuSetup = function (ltTarget, ltOwner, reason, behaviorParams){
    var ltPopupOwner = ..web.layout.element( behaviorParams.he )

}
```

layoutEleObject.onDataArrived

```
layoutEleObject.onDataArrived = function(ltTarget, ltOwner, data, dataType, status, url){
    return true;/*返回 true 撤消数据不显示内容*/
}
```

layoutEleObject.onDisabledStateChanged

```
layoutEleObject.onDisabledStateChanged = function (ltTarget, ltOwner, reason, behaviorParams) {

}
```

layoutEleObject.onDragEnter

```
layoutEleObject.onDragEnter = function( ltTarget, ltOwner, x, y, ltMouseParams ) {

```

```
}
```

layoutEleObject.onDragLeave

```
layoutEleObject.onDragLeave = function( ltTarget, ltOwner, x, y, ltMouseParams ) {

}
```

layoutEleObject.onDragRequest

```
layoutEleObject.onDragRequest = function( ltTarget, ltOwner, x, y, ltMouseParams ) {

}
```

layoutEleObject.onDrawBackground

```
layoutEleObject.onDrawBackground = function(ltOwner, hdc, rc){
    /*自绘背景*/
    return true;
}
```

layoutEleObject.onDrawContent

```
layoutEleObject.onDrawContent = function(ltOwner, hdc, rc){
    /*自绘内容*/
    return true;
}
```

layoutEleObject.onDrawForeground

```
layoutEleObject.onDrawForeground  = function(ltOwner, hdc, rc){
    /*自绘前景*/
    return true;
}
```

layoutEleObject.onDrop

```
layoutEleObject.onDrop = function( ltTarget, ltOwner, x, y, ltMouseParams ) {

}
```

layoutEleObject.onEditValueChanged

```
layoutEleObject.onEditValueChanged = function (ltTarget, ltOwner, reason, behaviorParams) {

}
```

layoutEleObject.onEditValueChanging

```
layoutEleObject.onEditValueChanging = function (ltTarget, ltOwner, reason, behaviorParams) {

}
```

layoutEleObject.onElementCollapsed

```
layoutEleObject.onElementCollapsed = function (ltTarget, ltOwner, reason, behaviorParams) {

}
```

layoutEleObject.onElementControlCreated

```
layoutEleObject.onElementControlCreated = function (ltTarget, ltOwner, reason, behaviorParams) {
    var ltCtrl = ltOwner.getCtrl();/*自定义控件已创建*/
}
```

layoutEleObject.onElementExpanded

```
layoutEleObject.onElementExpanded = function (ltTarget, ltOwner, reason, behaviorParams) {

}
```

layoutEleObject.onExchangeDrag

```
layoutEleObject.onExchangeDrag = function( ltTarget, ltOwner, x, y, cmd, dataTypes, fetchData, exParams ) {
    return true;/*这里返回 true 才会显示可拖放鼠标指针*/
}
```

layoutEleObject.onExchangeDragEnter

```
layoutEleObject.onExchangeDragEnter = function( ltTarget, ltOwner, x, y, cmd, dataTypes, fetchData, exParams ) {
    return true;
}
```

layoutEleObject.onExchangeDragLeave

```
layoutEleObject.onExchangeDragLeave = function( ltTarget, ltOwner, x, y, cmd, dataTypes, fetchData, exParams ) {
    return true;
}
```

layoutEleObject.onExchangeDrop

```
layoutEleObject.onExchangeDrop = function( ltTarget, ltOwner, x, y, cmd, dataTypes, fetchData, exParams ) {
    var data, dataType = fetchData( _HL_EXF/*可选指定支持的拖放数据类型*/ );
    if( data ) {
        return true;
    }
}
```

layoutEleObject.onFocusGot

```
layoutEleObject.onFocusGot = function( ltTarget, ltOwner, focusParams ) {

}
```

layoutEleObject.onFocusLost

```
layoutEleObject.onFocusLost = function( ltTarget, ltOwner, focusParams ) {
```

```
}
```

layoutEleObject.onFormReset

```
layoutEleObject.onFormReset = function (ltTarget, ltOwner, reason, behaviorParams) {

}
```

layoutEleObject.onFormSubmit

```
layoutEleObject.onFormSubmit = function (ltTarget, ltOwner, reason, behaviorParams) {
    var formData = behaviorParams.data.getValue();
}
```

layoutEleObject.onFrameDocumentComplete

```
layoutEleObject.onFrameDocumentComplete = function (ltTarget, ltOwner, reason, behaviorParams) {

}
```

layoutEleObject.onGesturePan

```
layoutEleObject.onGesturePan = function (ltTarget, ltOwner, ltGestureParams) {
    /*平移*/
}
```

layoutEleObject.onGestureRequest

```
layoutEleObject.onGestureRequest = function (ltTarget, ltOwner, ltGestureParams) {
    ltGestureParams.flags = 0xFFFF/*_HL_GESTURE_FLAGS_ALL*/
    return true;
}
```

layoutEleObject.onGestureRotate

```
layoutEleObject.onGestureRotate = function (ltTarget, ltOwner, ltGestureParams) {
    /*旋转*/
}
```

layoutEleObject.onGestureTap1

```
layoutEleObject.onGestureTap1 = function (ltTarget, ltOwner, ltGestureParams) {
    /*单击*/
}
```

layoutEleObject.onGestureTap2

```
layoutEleObject.onGestureTap2 = function (ltTarget, ltOwner, ltGestureParams) {
    /*双击*/
}
```

layoutEleObject.onGestureZoom

```
layoutEleObject.onGestureZoom = function (ltTarget, ltOwner, ltGestureParams) {
    /*缩放*/
}
```

layoutEleObject.onGetCaretPos

```
layoutEleObject.onSetValue = function(  ltOwner, value ){

    return true, /*光标位置:left, top, width, height*/
}
```

layoutEleObject.onGetValue

```
layoutEleObject.onGetValue = function( ltOwner ){

    return true, /*返回控件值*/;
}
```

layoutEleObject.onHyperlinkClick

```
layoutEleObject.onHyperlinkClick = function (ltTarget, ltOwner, reason, behaviorParams) {

}
```

layoutEleObject.onInitDataView

```
layoutEleObject.onInitDataView = function (ltTarget, ltOwner, reason, behaviorParams) {

}
```

layoutEleObject.onIsEmpty

```
layoutEleObject.onSetValue = function(  ltOwner, value ){

    return true
}
```

layoutEleObject.onKeyChar

```
layoutEleObject.onKeyChar = function( ltTarget, ltOwner, keyCode,  altState,  ltKeyParams ) {

}
```

layoutEleObject.onKeyDown

```
layoutEleObject.onKeyDown = function( ltTarget, ltOwner, keyCode,  altState,  ltKeyParams ) {

}
```

layoutEleObject.onKeyUp

```
layoutEleObject.onKeyUp = function( ltTarget, ltOwner, keyCode,  altState,  ltKeyParams ) {
```

```
}
```

layoutEleObject.onMenuItemActive

```
layoutEleObject.onMenuItemActive = function (ltTarget, ltOwner, reason, behaviorParams) {
    var ltPopupOwner = ..web.layout.element( behaviorParams.he )

}
```

layoutEleObject.onMenuItemClick

```
layoutEleObject.onMenuItemClick = function (ltTarget, ltOwner, reason, behaviorParams) {
    var ltPopupOwner = ..web.layout.element( behaviorParams.he )

}
```

layoutEleObject.onMouseClick

```
layoutEleObject.onMouseClick = function( ltTarget, ltOwner, x, y, ltMouseParams ) {

}
```

layoutEleObject.onMouseDblClick

```
layoutEleObject.onMouseDblClick = function( ltTarget, ltOwner, x, y, ltMouseParams ) {

}
```

layoutEleObject.onMouseDown

```
layoutEleObject.onMouseDown = function( ltTarget, ltOwner, x, y, ltMouseParams ) {

}
```

layoutEleObject.onMouseEnter

```
layoutEleObject.onMouseEnter = function( ltTarget, ltOwner, x, y, ltMouseParams ) {

}
```

layoutEleObject.onMouseIdle

```
layoutEleObject.onMouseIdle = function( ltTarget, ltOwner, x, y, ltMouseParams ) {

}
```

layoutEleObject.onMouseLeave

```
layoutEleObject.onMouseLeave = function( ltTarget, ltOwner, x, y, ltMouseParams ) {

}
```

layoutEleObject.onMouseMove

```
layoutEleObject.onMouseMove = function( ltTarget, ltOwner, x, y, ltMouseParams ) {

}
```

layoutEleObject.onMouseTick

```
layoutEleObject.onMouseTick = function( ltTarget, ltOwner, x, y, ltMouseParams ) {

}
```

layoutEleObject.onMouseUp

```
layoutEleObject.onMouseUp = function( ltTarget, ltOwner, x, y, ltMouseParams ) {

}
```

layoutEleObject.onMouseWheel

```
layoutEleObject.onMouseWheel = function( ltTarget, ltOwner, x, y, ltMouseParams ) {

}
```

layoutEleObject.onPopupDismissed

```
layoutEleObject.onPopupDismissed = function (ltTarget, ltOwner, reason, behaviorParams) {
    var ltPopupOwner = ..web.layout.element( behaviorParams.he )

}
```

layoutEleObject.onPopupDismissing

```
layoutEleObject.onPopupDismissing = function (ltTarget, ltOwner, reason, behaviorParams) {

}
```

layoutEleObject.onPopupReady

```
layoutEleObject.onPopupReady = function (ltTarget, ltOwner, reason, behaviorParams) {
    var ltPopupOwner = ..web.layout.element( behaviorParams.he )

}
```

layoutEleObject.onPopupRequest

```
layoutEleObject.onPopupRequest = function (ltTarget, ltOwner, reason, behaviorParams) {
    var ltPopupOwner = ..web.layout.element( behaviorParams.he )

}
```

layoutEleObject.onReplaceSel

```
layoutEleObject.onSetValue = function(  ltOwner, value ){
```

```
    return true
}
```

layoutEleObject.onRequestTooltip

```
layoutEleObject.onRequestTooltip = function (ltTarget, ltOwner, reason, behaviorParams) {

}
```

layoutEleObject.onRowsDataRequest

```
layoutEleObject.onRowsDataRequest = function (ltTarget, ltOwner, reason, behaviorParams) {

}
```

layoutEleObject.onSelectSelectionChanged

```
layoutEleObject.onSelectSelectionChanged = function (ltTarget, ltOwner, reason, behaviorParams) {
    var ltOption = ..web.layout.element( behaviorParams.he )

}
```

layoutEleObject.onSelectStateChanged

```
layoutEleObject.onSelectStateChanged = function (ltTarget, ltOwner, reason, behaviorParams) {

}
```

layoutEleObject.onSetValue

```
layoutEleObject.onSetValue = function(  ltOwner, value ){
    return true
}
```

layoutEleObject.onSize

```
layoutEleObject.onSize = function (ltOwner) {

}
```

layoutEleObject.onTableHeaderClick

```
layoutEleObject.onTableHeaderClick = function (ltTarget, ltOwner, cellIndex, behaviorParams) {

}
```

layoutEleObject.onTableRowClick

```
layoutEleObject.onTableRowClick = function (ltTarget, ltOwner, rowIndex, behaviorParams) {

}
```

layoutEleObject.onTableRowDblClick

```
layoutEleObject.onTableRowDblClick = function (ltTarget, ltOwner, rowIndex, behaviorParams) {

}
```

layoutEleObject.onTimer

```
layoutEleObject.onTimer = function (ltOwner, timerId) {

}
```

layoutEleObject.onUiStateChanged

```
layoutEleObject.onUiStateChanged = function (ltTarget, ltOwner, reason, behaviorParams) {

}
```

layoutEleObject.onVisualStateChanged

```
layoutEleObject.onVisualStateChanged = function (ltTarget, ltOwner, shown, behaviorParams) {

}
```

layoutEleObject.outerHTML

节点 HTML 源码

layoutEleObject.parent()

获取父节点

返回对象:layoutEleObject

layoutEleObject.popup

在指定坐标弹出菜单

layoutEleObject.popup(x 坐标，y 坐标，是否显示动画，相应坐标)

参数三，四可省略，参数四可选值为 1 到 9，表示相对指定坐标的偏移位置如下:

3 2 1

6 5 4

9 8 7

layoutEleObject.popupAnchor(锚节点，选项)

弹出菜单节点

参数 2 指定弹出方向，可省略，默认为 2

可选值(2:下，4:左;6:右 8:上)

8

4 5 6

2

layoutEleObject.popupHide()

隐藏菜单节点

layoutEleObject.popupTrack

显示菜单、并等待点击返回用户点选的菜单项

layoutEleObject.popupTrack(x 坐标，y 坐标，是否显示动画，相应坐标)

参数三，四可省略，参数四可选值为 1 到 9，表示相对指定坐标的偏移位置如下:

3 2 1

6 5 4

9 8 7

layoutEleObject.postEvent(事件 ID，选项，ltTarget)

触发事件，不等待返回，参数二，参数三为可选参数

参数一也可以是事件函数名，例如"onButtonClick"，支持 web.layout.event.BEHAVIOR 名字空间的事件，或其他 on 前缀的自定义事件，不应使用此函数触发其他例如 onMuse，onKey 前缀的事件

layoutEleObject.previous()

上一个兄弟节点

返回对象:layoutEleObject

layoutEleObject.previousSibling()

获取上一个兄弟节点，可选在参数中指定偏移值

返回对象:layoutEleObject

layoutEleObject.printf("格式化串"，其他参数)

格式化并写入 HTML 代码到节点尾部

layoutEleObject.queryElement()

返回对象:layoutEleObject

layoutEleObject.queryElements()

返回对象:layoutEleObject

layoutEleObject.queryParent("标签名[属性名='属性值']")

该函数类似于 CSSS!脚本中的$1p 函数，不同的是会测试自身是否匹配，使用 CSS 选择器语法，在父节点中查找匹配最近的一个父节点，可选使用参数二指定向上搜索深度，如果搜索级别省略或设为 0，则不限制搜索深度，如果该值为 1，仅匹配自身

layoutEleObject.queryParent()

返回对象:layoutEleObject

layoutEleObject.queryParents("标签名[属性名='属性值']")

该函数类似于 CSSS!脚本中的$p 函数，不同的是会测试自身是否匹配，使用 CSS 选择器语法，在父节点中查找所有匹配的节点，成功返回非空数组

layoutEleObject.querySelector

使用 CSS 选择器语法查找节点，返回节点对象

layoutEleObject.querySelector("标签名[属性名='属性值']")

使用 CSS 选择器语法，在当前节点的子节点中查找第一个符合条件节点，如果有多个参数则首先调用 string.format 格式化为 CSS 文本，该函数名可使用'$1'代替，等价于 CSS!中的$1 函数

layoutEleObject.querySelector()

返回对象:layoutEleObject

layoutEleObject.querySelectorAll

使用 CSS 选择器语法查找节点，返回数组

layoutEleObject.querySelectorAll("标签名[属性名='属性值']")

使用 CSS 选择器语法，在当前节点的子节点中查找所有符合条件节点，如果有多个参数则首先调用 string.format 格式化为 CSS 文本，该函数名可使用'$'代替，等价于 CSS!中的$函数

layoutEleObject.querySelectorAll()

返回对象:layoutEleObject

layoutEleObject.release()

确认节点对象不再使用可使用此函数释放引用，调用此函数后不可再使用该对象，节点会自动释放，不建议手动调用此函数

layoutEleObject.request(网址)

用于框架节点打开网页
该函数会触发 onDataArrived 事件

layoutEleObject.request(网址，参数，提交方法)

参数可以是字符串或键值对组成的表对象
提交方法可省略，默认为"POST"
该函数可触发 onDataArrived 事件

layoutEleObject.root()

根文档的根节点
获取框架内部当前文档根节点应请使用 documentElement 属性
返回对象:layoutEleObject

layoutEleObject.scrollToView(滚动到顶部，平滑效果)

滚动上层视图节点滚动条

layoutEleObject.selHTML

HTML 剪帖格式返回选区内容，该属性只读

layoutEleObject.selText

获取或设置选区文本

layoutEleObject.sendEvent(事件 ID，选项，ltTarget)

触发事件，并等待返回，参数二，参数三为可选参数
参数一也可以是事件函数名，例如"onButtonClick"，支持 web.layout.event.BEHAVIOR 名字空间的事件，或其他 on 前缀的自定义事件，不应使用此函数触发其他例如 onMuse，onKey 前缀的事件

layoutEleObject.setAttribute("hide"，"true")

修改属性，属性值只能是字符串或 null

也可以直接写 ele.属性名字 = "值"

layoutEleObject.setDataTable(tab 对象)

序列化 table 对象并设置为节点 data-table 属性

layoutEleObject.setEventRoot()

设为事件触发根节点

节点之外的节点事节被禁用，成功返回上一个事件根节点

layoutEleObject.setPos(x 坐标，y 坐标，宽，高)

移动节点，所有参数可选

layoutEleObject.setScrollPos(x，y，平滑效果)

x，y 皆可省略，默认值为 0

layoutEleObject.setTimer(延时毫秒值，ID)

如果延时值为 0，取消定时器，ID 可省略

在 CSSS!脚本中可触发 timer! 事件

layoutEleObject.setValueObject(值，单位，类型)

值可选使用字符串，数值，时间值，或 web.layout.valueObject 对象，单位为可选参数，值类型(使用 HL_T 前缀常量表示)为可选参数

layoutEleObject.setsel(起始位置，结束位置)

设置文本框控件文本选区

layoutEleObject.sort(cmpFunc，firstInddex，lastIndex)

```
layoutEleObject.sort(
    function(ltEle，ltEle2){

        return /*比较条件，可返回-1，0，1 等值表示比较结果*/ ？ -1 ： 1;
    }
)
```

layoutEleObject.state

节点状态，类似于 CSSS! 脚本中使用一个冒号表示状态，例如 ele:hover = false，

返回对象:htmlayoutStateObject

layoutEleObject.style

样式

类似于 CSSS! 脚本中使用一对冒号表示状态，例如 ele::width = 56px，

返回对象:htmlayoutStyleObject

layoutEleObject.swap(节点对象)

交换节点对象位置

layoutEleObject.tagName

返回表示节点类型的字符串
例如层节点返回"div"

layoutEleObject.test("标签名[属性名='属性值']"，是否测试所有父节点)

测试节点自身是否符合匹配，该函数等价于 testParent(CSS 选择器，1)

layoutEleObject.testParent("标签名[属性名='属性值']")

测试父节点中是否存在符合匹配的节点，可使用参数 2 指定向上搜索深度，默认为 0 表示不作限制，

layoutEleObject.traverseKey("onKeyDown"，_VK_...，altSate)

参数@1 指定事件，可直接使用 HL_KEY 前缀常量作为参数
参数@2 指定虚拟键码，参数 altState 可省略，可选值:
_HL_CONTROL_KEY_PRESSED:0x1，_HL_SHIFT_KEY_PRESSED:0x2，_HL_ALT_KEY_PRESSED:0x4

layoutEleObject.traverseMouse("onMouseClick"，x，y，btnState，altSate)

除参数@1 以外，其他都是可选参数
参数@1 指定事件，可直接使用 HL_MOUSE 前缀常量作为参数
x，y 用于指定节点内部相对坐标
btnState 默认为 _HL_MAIN_MOUSE_BUTTON，即鼠标左键

layoutEleObject.type

返回控件类型(type 属性)

layoutEleObject.uid

节点唯一标志符(数值)

layoutEleObject.update()

更新;

layoutEleObject.update(true)

更新，
重新计算元素的尺寸

layoutEleObject.updateEx(_HL_UPDATE_ELEMENT_FLAGS__)

更新;
并调用::UpdateWindow 刷新窗口

layoutEleObject.updateHTML()

重新解析节点 HTML 代码

layoutEleObject.updateWindow(是否顶层窗口)

更新窗口
参数可省略，默认为 false

layoutEleObject.value

读写控件值，可选值类型:字符串值，数值，时间值等，可选使用 web.layout.value 对象赋值

layoutEleObject.visitElement()

返回对象:layoutEleObject

layoutEleObject.visitElement(节点类型，属性名，属性值，深度)

查找第一个符合的节点，所有参数都可空

layoutEleObject.visitElements(节点类型，属性名，属性值，深度)

查找所有符合的节点，所有参数都可空

layoutEleObject.write(html，...)

写入 HTML 并替换节点内部 HTML，参数可以是数值，字符串

layoutEleObject.xcall("自定义函数名"，其他参数...)

调用节点自定义的函数，并获取返回值

ltEleScrollInfoObject 成员列表

ltEleScrollInfoObject.page

分页大小

返回对象:sizeObject

ltEleScrollInfoObject.pos

滚动条位置

返回对象:pointObject

ltEleScrollInfoObject.rect

视图区块

返回对象:rectObject

ltEleScrollInfoObject.size

内容大小

返回对象:sizeObject

ltOwner 成员列表

HTMLayout 节点对象，在事件函数中，ltEle 一般表示捕获事件的节点，lt 前缀变量为 HTMLayout 保留，请勿用于表示其他类型对象

返回对象:layoutEleObject

web.layout 成员列表

web.layout.element()

返回对象:layoutEleObject

web.layout.element(句柄，是否添加引用)

将节点句柄转换为节点对象

自动完成常量

_HLDOM_INVALID_HANDLE=2
_HLDOM_INVALID_HWND=1
_HLDOM_INVALID_PARAMETER=4
_HLDOM_OK=0
_HLDOM_OK_NOT_HANDLED=-1
_HLDOM_OPERATION_FAILED=5
_HLDOM_PASSIVE_HANDLE=3
_HL_BACK_IMAGE_AREA=0x40
_HL_BORDER_BOX=0x20
_HL_CONTAINER_RELATIVE=3
_HL_CONTENT_BOX=0
_HL_CTL_BUTTON=4
_HL_CTL_CALENDAR=0x16
_HL_CTL_CHECKBOX=5
_HL_CTL_CURRENCY=0x10
_HL_CTL_DATE=0x17
_HL_CTL_DD_SELECT=9
_HL_CTL_DECIMAL=0xF
_HL_CTL_EDIT=2
_HL_CTL_FORM=0x23
_HL_CTL_FRAME=0x19
_HL_CTL_FRAMESET=0x1A
_HL_CTL_GRAPHICS=0x1B
_HL_CTL_HIDDEN=0x20
_HL_CTL_HTMLAREA=0xB
_HL_CTL_HYPERLINK=0x12
_HL_CTL_LIST=0x1D
_HL_CTL_MENU=0x14
_HL_CTL_MENUBAR=0x13
_HL_CTL_MENUBUTTON=0x15
_HL_CTL_NO=0
_HL_CTL_NUMERIC=3
_HL_CTL_PASSWORD=0xC
_HL_CTL_PROGRESS=0xD
_HL_CTL_RADIO=6
_HL_CTL_RICHTEXT=0x1E
_HL_CTL_SCROLLBAR=0x11
_HL_CTL_SELECT_MULTIPLE=8
_HL_CTL_SELECT_SINGLE=7
_HL_CTL_SLIDER=0xE
_HL_CTL_SPRITE=0x1C
_HL_CTL_TEXTAREA=0xA
_HL_CTL_TIME=0x18
_HL_CTL_TOOLBAR=0x22
_HL_CTL_TOOLTIP=0x1F
_HL_CTL_UNKNOWN=1
_HL_CTL_URL=0x21
_HL_FORE_IMAGE_AREA=0x50
_HL_HV_BAD_PARAMETER=1
_HL_HV_INCOMPATIBLE_TYPE=2
_HL_HV_OK=0
_HL_HV_OK_TRUE=-1
_HL_MARGIN_BOX=0x30
_HL_PADDING_BOX=0x10
_HL_ROOT_RELATIVE=1
_HL_SCROLLABLE_AREA=0x60
_HL_SELF_RELATIVE=2
_HL_T_ARRAY=9
_HL_T_BOOL=2
_HL_T_BYTES=0xC
_HL_T_CURRENCY=7
_HL_T_DATE=6
_HL_T_DOM_OBJECT=0xE
_HL_T_FLOAT=4
_HL_T_FUNCTION=0xB
_HL_T_INT=3
_HL_T_LENGTH=8
_HL_T_MAP=0xA
_HL_T_NULL=1
_HL_T_OBJECT=0xD
_HL_T_STRING=5
_HL_T_UNDEFINED=0
_HL_UPDATE_ELEMENT_FLAGS_MEASURE_DEEP=2
_HL_UPDATE_ELEMENT_FLAGS_MEASURE_INPLACE=1
_HL_UPDATE_ELEMENT_FLAGS_REDRAW_NOW=0x8000
_HL_UPDATE_ELEMENT_FLAGS_RESET_STYLE_DEEP=0x10
_HL_UPDATE_ELEMENT_FLAGS_RESET_STYLE_THIS=0x20
_HL_UT_CM=9
_HL_UT_COLOR=0xF
_HL_UT_DIP=0xD
_HL_UT_EM=1
_HL_UT_EX=2

_HL_UT_IN=8
_HL_UT_MM=0xA
_HL_UT_PC=0xC
_HL_UT_PR=3
_HL_UT_PT=0xB
_HL_UT_PX=7
_HL_UT_SP=4
_HL_UT_SYMBOL=0xFFFF
_HL_UT_URL=0x10
_HL_VIEW_RELATIVE=4

4、web.layout.element.state

htmlayoutStateObject 成员列表

htmlayoutStateObject.active

鼠标按下状态

htmlayoutStateObject.anchor

多选的选区端点

htmlayoutStateObject.animating

是否正在播放动画

htmlayoutStateObject.busy

节点忙状态
指正在下载、但尚未下载结束
CSS 中 frame:not(:busy):incomplete 匹配框架下载失败数据未加载

htmlayoutStateObject.checked

是否选中

htmlayoutStateObject.collapsed

折叠状态

htmlayoutStateObject.copying

是否正在拖动,复制模式

htmlayoutStateObject.current

当前选中节点

htmlayoutStateObject.disabled

禁用状态

htmlayoutStateObject.dragOver

是否正在当前节点上拖动
CSS 中对应 :drag-over

htmlayoutStateObject.dragSource

是否拖放源节点
CSS 中对应 :drag-source

htmlayoutStateObject.dropMarker

是否指示拖放目的位置的副本

CSS 中对应 :drag-marker

htmlayoutStateObject.dropTarget

是否拖放目标节点

CSS 中对应 :drag-target

htmlayoutStateObject.empty

文本内容为空

htmlayoutStateObject.expanded

展开状态

htmlayoutStateObject.focus

拥有焦点

htmlayoutStateObject.focusable

可接受输入焦点;

htmlayoutStateObject.hover

鼠标悬停

htmlayoutStateObject.incomplete

背景或前景图像、框架页面等请求的资源未加载完成

CSS 中 frame:not(:busy):incomplete 匹配框架下载失败数据未加载

htmlayoutStateObject.link

未被点击超链接

htmlayoutStateObject.ltr

是否左向右显示

htmlayoutStateObject.moving

是否正在拖动,移动模式

htmlayoutStateObject.ownsPopup

是否弹出节点的所有者节点

CSS 中对应 :owns-popup

htmlayoutStateObject.popup

弹出状态

htmlayoutStateObject.pressed

键按下

htmlayoutStateObject.readonly

只读

htmlayoutStateObject.rtl

是否右向左显示

htmlayoutStateObject.synthetic

是否引擎自动生成的节点

htmlayoutStateObject.tabfocus

通过 tab 键得到焦点

htmlayoutStateObject.visited

点击过的超链接

自动完成常量

_HL_STATE_ACTIVE=4
_HL_STATE_ANCHOR=0x4000
_HL_STATE_ANIMATING=0x1000
_HL_STATE_BUSY=0x80000
_HL_STATE_CHECKED=0x40
_HL_STATE_COLLAPSED=0x400
_HL_STATE_COPYING=0x800000
_HL_STATE_CURRENT=0x20
_HL_STATE_DISABLED=0x80
_HL_STATE_DRAG_OVER=0x100000
_HL_STATE_DRAG_SOURCE=0x1000000
_HL_STATE_DROP_MARKER=0x2000000
_HL_STATE_DROP_TARGET=0x200000
_HL_STATE_EMPTY=0x40000
_HL_STATE_EXPANDED=0x200
_HL_STATE_FOCUS=8
_HL_STATE_FOCUSABLE=0x2000
_HL_STATE_HOVER=2
_HL_STATE_INCOMPLETE=0x800
_HL_STATE_IS_LTR=0x10000000
_HL_STATE_IS_RTL=0x20000000
_HL_STATE_LINK=1
_HL_STATE_MOVING=0x400000
_HL_STATE_OWNS_POPUP=0x10000
_HL_STATE_POPUP=0x8000000
_HL_STATE_PRESSED=0x4000000
_HL_STATE_READONLY=0x100
_HL_STATE_SYNTHETIC=0x8000
_HL_STATE_TABFOCUS=0x20000
_HL_STATE_VISITED=0x10

5、web.layout.element.style

htmlayoutStyleObject 成员列表

htmlayoutStyleObject.cursor

鼠标指针，值必须是字符串或 null

htmlayoutStyleObject.display

显示样式，值必须是字符串或 null

htmlayoutStyleObject.height

高度，值必须是字符串或 null

htmlayoutStyleObject.left

左侧位置，值必须是字符串或 null

htmlayoutStyleObject.margin

外边距，值必须是字符串或 null

htmlayoutStyleObject.padding

内边距，值必须是字符串或 null

htmlayoutStyleObject.top

顶部位置，值必须是字符串或 null

htmlayoutStyleObject.visiblity

可见状态，值必须是字符串或 null

htmlayoutStyleObject.width

宽度，值必须是字符串或 null

6、web.layout.valueObject

web.layout 成员列表

web.layout.valueObject()

返回对象:webLayoutValueObject

web.layout.valueObject(初始值，单位，类型)

创建并返回动态值类型，所有参数可选

web.layout.valueObjectLite()

无 GC 值对象，避免自动释放

返回对象:webLayoutValueObject

webLayoutValueObject 成员列表

webLayoutValueObject.addRef()

添加引用计数

webLayoutValueObject.clear()

清空值

webLayoutValueObject.clone()

复制值

返回对象:webLayoutValueObject

webLayoutValueObject.enum(枚举函数)

```
webLayoutValueObject.enum(
    function(k，v ){
        io.print( k，v )
    }
)
```

webLayoutValueObject.getEle()

返回节点对象，注意:节点类型只能读取不能写入

webLayoutValueObject.getInt32()

返回整型数值

如果值类型不是整型值，取值返回空

webLayoutValueObject.getLong64()

返回长整型数值

如果值类型不是长整型数值，取值返回空

webLayoutValueObject.getNumber()

返回浮点数值

如果值类型不是浮点数，取值返回空

使用 tonumber() 函数可强制返回数值

webLayoutValueObject.getString()

返回字符串值

如果值类型字符串值，取值返回空

使用 tostring() 函数可强制返回数值

webLayoutValueObject.getTime()

返回时间值

如果值类型不是时间值，取值返回空

webLayoutValueObject.getType()

返回值类型 HL_T 前缀常量表示

webLayoutValueObject.getValue()

返回字符串值、数值、时间值

webLayoutValueObject.isArray()

是否数组对象

webLayoutValueObject.isDomObject()

是否节点对象，

使用 getEle()函数转换为节点对象

webLayoutValueObject.isMap()

是否键值表对象

webLayoutValueObject.jsonParse('{ k:"v" }')

解析 JSON 字符串

webLayoutValueObject.jsonStringify()

转换并返回 JSON 字符串

webLayoutValueObject.length

数组长度

webLayoutValueObject.parse("10pt")

解析字符串

webLayoutValueObject.setInt32(__/*数值*/)

写入整型数值

webLayoutValueObject.setLong64(__/*数值*/)

写入长整型数值

webLayoutValueObject.setNumber(__/*数值*/)

写入浮点数值

webLayoutValueObject.setString(__/*字符串值*/)

写入字符串值

webLayoutValueObject.setTime(__/*时间值*/)

写入时间值

webLayoutValueObject.setValue(值，单位，类型)

写入字符串值、数值、时间值

参数 2，参数 3 为可选参数

webLayoutValueObject.stringify()

转换并返回字符串

webLayoutValueObject.value

读取或写入字符串值、数值、时间值

自动完成常量

_HL_CVT_JSON_LITERAL=1

_HL_CVT_JSON_MAP=2

_HL_CVT_SIMPLE=0

7、web.layout.event

web.layout.event 成员列表

web.layout.event.KEY_PARAMS()

创建鼠标事件结构体参数

返回对象:hlKeyParamsObject

web.layout.event.MOUSE_PARAMS()

创建鼠标事件结构体参数

返回对象:hlMouseParamsObject

hlKeyParamsObject 成员列表

hlKeyParamsObject.alt_state

控制键状态

hlKeyParamsObject.cmd

命令 ID

hlKeyParamsObject.key_code

虚拟键码

hlKeyParamsObject.pointer

结构体指针地址

hlKeyParamsObject.target

触发事件节点

hlMouseParamsObject 成员列表

hlMouseParamsObject.alt_state

控制键状态

hlMouseParamsObject.button_state

按键状态

在 onMouseWheel 事件中表示滚轮数据

hlMouseParamsObject.cmd

命令 ID

hlMouseParamsObject.cursor_type

指针

hlMouseParamsObject.dragging

正在被拖放的节点，节点要在 CSS 属性 draggable 中启用拖动

该节点不为空值是，isdragging 为真

hlMouseParamsObject.dragging_mode

拖动类型，_HL_DRAGGING_MOVE 或 _HL_DRAGGING_COPY

hlMouseParamsObject.is_on_icon

是否位于图标上，这里指的是前景图片，并具有 no-repeat 样式

hlMouseParamsObject.isdragging

是否正在拖动，无论节点 CSS 是否指定 draggable 属性

hlMouseParamsObject.pointer

结构体指针地址

hlMouseParamsObject.pos

节点坐标

返回对象:pointObject

hlMouseParamsObject.pos_document

文档坐标

返回对象:pointObject

hlMouseParamsObject.target

触发事件节点

weblayoutObject 成员列表

weblayoutObject.onActiveChild

```
weblayoutObject.onActiveChild = function (ltTarget, ltOwner, reason, behaviorParams) {

}
```

weblayoutObject.onAnimation

```
weblayoutObject.onAnimation = function (ltTarget, ltOwner, reason, behaviorParams) {

}
```

weblayoutObject.onApplicationEvent

```
weblayoutObject.onApplicationEvent = function (ltTarget, ltOwner, reason, behaviorParams) {
    /*postEvent 或 sendEvent 发出一大于 0x100 的消息*/
}
```

weblayoutObject.onAttach

```
weblayoutObject.onAttach = function( ltOwner ){

    return true
```

```
}
```

weblayoutObject.onButtonClick

```
weblayoutObject.onButtonClick = function (ltTarget, ltOwner, reason, behaviorParams) {

}
```

weblayoutObject.onButtonPress

```
weblayoutObject.onButtonPress = function (ltTarget, ltOwner, reason, behaviorParams) {

}
```

weblayoutObject.onButtonStateChanged

```
weblayoutObject.onButtonStateChanged = function (ltTarget, ltOwner, reason, behaviorParams) {

}
```

weblayoutObject.onClosePopup

```
weblayoutObject.onClosePopup = function (ltTarget, ltOwner, reason, behaviorParams) {

}
```

weblayoutObject.onContextMenuRequest

```
weblayoutObject.onContextMenuRequest = function (ltTarget, ltOwner, reason, behaviorParams) {
    var ltPopupOwner = ..web.layout.element( behaviorParams.he )

}
```

weblayoutObject.onContextMenuSetup

```
weblayoutObject.onContextMenuSetup = function (ltTarget, ltOwner, reason, behaviorParams) {
    var ltPopupOwner = ..web.layout.element( behaviorParams.he )

}
```

weblayoutObject.onDataArrived

```
weblayoutObject.onDataArrived = function(ltTarget, ltOwner, data, dataType, status, url){
    return true;/*返回 true 撤消数据不显示内容*/
}
```

weblayoutObject.onDetach

```
weblayoutObject.onDetach = function( ltOwner ){

    return true
}
```

weblayoutObject.onDisabledStateChanged

```
weblayoutObject.onDisabledStateChanged = function (ltTarget, ltOwner, reason, behaviorParams) {

}
```

weblayoutObject.onDragEnter

```
weblayoutObject.onDragEnter = function (ltTarget, ltOwner, x, y, ltMouseParams) {

}
```

weblayoutObject.onDragLeave

```
weblayoutObject.onDragLeave = function (ltTarget, ltOwner, x, y, ltMouseParams) {

}
```

weblayoutObject.onDragRequest

```
weblayoutObject.onDragRequest = function (ltTarget, ltOwner, x, y, ltMouseParams) {

}
```

weblayoutObject.onDrawBackground

```
weblayoutObject.onDrawBackground = function(ltOwner, hdc, rc){

}
```

weblayoutObject.onDrawContent

```
weblayoutObject.onDrawContent = function(ltOwner, hdc, rc){

}
```

weblayoutObject.onDrawForeground

```
weblayoutObject.onDrawForeground = function(ltOwner, hdc, rc){

}
```

weblayoutObject.onDrop

```
weblayoutObject.onDrop = function (ltTarget, ltOwner, x, y, ltMouseParams) {

}
```

weblayoutObject.onEditValueChanged

```
weblayoutObject.onEditValueChanged = function (ltTarget, ltOwner, reason, behaviorParams) {

}
```

weblayoutObject.onEditValueChanging

```
weblayoutObject.onEditValueChanging = function (ltTarget, ltOwner, reason, behaviorParams) {

}
```

weblayoutObject.onElementCollapsed

```
weblayoutObject.onElementCollapsed = function (ltTarget, ltOwner, reason, behaviorParams) {

}
```

weblayoutObject.onElementControlCreated

```
weblayoutObject.onElementControlCreated = function (ltTarget, ltOwner, reason, behaviorParams) {
    var ltCtrl = ltOwner.getCtrl();/*自定义控件已创建*/
}
```

weblayoutObject.onElementExpanded

```
weblayoutObject.onElementExpanded = function (ltTarget, ltOwner, reason, behaviorParams) {

}
```

weblayoutObject.onExchangeDrag

```
weblayoutObject.onExchangeDrag = function( ltTarget, ltOwner, x, y, cmd, dataTypes, fetchData, exParams ) {
    return true;/*这里返回 true 才会显示可拖放鼠标指针*/
}
```

weblayoutObject.onExchangeDragEnter

```
weblayoutObject.onExchangeDragEnter = function( ltTarget, ltOwner, x, y, cmd, dataTypes, fetchData, exParams ) {
    return true;
}
```

weblayoutObject.onExchangeDragLeave

```
weblayoutObject.onExchangeDragLeave = function( ltTarget, ltOwner, x, y, cmd, dataTypes, fetchData, exParams ) {
    return true;
}
```

weblayoutObject.onExchangeDrop

```
weblayoutObject.onExchangeDrop = function( ltTarget, ltOwner, x, y, cmd, dataTypes, fetchData, exParams ) {
    var data, dataType = fetchData( _HL_EXF/*可选指定支持的拖放数据类型*/ );
    if( data ) {
        return true;
    }
}
```

weblayoutObject.onFocusGot

```
weblayoutObject.onFocusGot = function (ltTarget, ltOwner, focusParams) {

}
```

weblayoutObject.onFocusLost

```
weblayoutObject.onFocusLost = function (ltTarget, ltOwner, focusParams) {

}
```

weblayoutObject.onFormReset

```
weblayoutObject.onFormReset = function (ltTarget, ltOwner, reason, behaviorParams) {

}
```

weblayoutObject.onFormSubmit

```
weblayoutObject.onFormSubmit = function (ltTarget, ltOwner, reason, behaviorParams) {
    var formData = behaviorParams.data.getValue();
}
```

weblayoutObject.onFrameDocumentComplete

```
weblayoutObject.onFrameDocumentComplete = function (ltTarget, ltOwner, reason, behaviorParams) {

}
```

weblayoutObject.onGesturePan

```
weblayoutObject.onGesturePan = function (ltTarget, ltOwner, ltGestureParams) {
    /*平移*/
}
```

weblayoutObject.onGestureRequest

```
weblayoutObject.onGestureRequest = function (ltTarget, ltOwner, ltGestureParams) {
    ltGestureParams.flags = 0xFFFF/*_HL_GESTURE_FLAGS_ALL*/
    return true;
}
```

weblayoutObject.onGestureRotate

```
weblayoutObject.onGestureRotate = function (ltTarget, ltOwner, ltGestureParams) {
    /*旋转*/
}
```

weblayoutObject.onGestureTap1

```
weblayoutObject.onGestureTap1 = function (ltTarget, ltOwner, ltGestureParams) {
    /*单击*/
}
```

weblayoutObject.onGestureTap2

```
weblayoutObject.onGestureTap2 = function (ltTarget, ltOwner, ltGestureParams) {
    /*双击*/
}
```

weblayoutObject.onGestureZoom

```
weblayoutObject.onGestureZoom = function (ltTarget, ltOwner, ltGestureParams) {
    /*缩放*/
}
```

weblayoutObject.onHyperlinkClick

```
weblayoutObject.onHyperlinkClick = function (ltTarget, ltOwner, reason, behaviorParams) {

}
```

weblayoutObject.onInitDataView

```
weblayoutObject.onInitDataView = function (ltTarget, ltOwner, reason, behaviorParams) {

}
```

weblayoutObject.onKeyChar

```
weblayoutObject.onKeyChar = function (ltTarget, ltOwner, keyCode,  altState,  ltKeyParams) {

}
```

weblayoutObject.onKeyDown

```
weblayoutObject.onKeyDown = function (ltTarget, ltOwner, keyCode,  altState,  ltKeyParams) {

}
```

weblayoutObject.onKeyUp

```
weblayoutObject.onKeyUp = function (ltTarget, ltOwner, keyCode,  altState,  ltKeyParams) {

}
```

weblayoutObject.onMenuItemActive

```
weblayoutObject.onMenuItemActive = function (ltTarget, ltOwner, reason, behaviorParams) {
    var ltPopupOwner = ..web.layout.element( behaviorParams.he )

}
```

weblayoutObject.onMenuItemClick

```
weblayoutObject.onMenuItemClick = function (ltTarget, ltOwner, reason, behaviorParams) {
    var ltPopupOwner = ..web.layout.element( behaviorParams.he )
```

```
}
```

weblayoutObject.onMouseClick

```
weblayoutObject.onMouseClick = function (ltTarget, ltOwner, x, y, ltMouseParams) {

}
```

weblayoutObject.onMouseDblClick

```
weblayoutObject.onMouseDblClick = function (ltTarget, ltOwner, x, y, ltMouseParams) {

}
```

weblayoutObject.onMouseDown

```
weblayoutObject.onMouseDown = function (ltTarget, ltOwner, x, y, ltMouseParams) {

}
```

weblayoutObject.onMouseEnter

```
weblayoutObject.onMouseEnter = function (ltTarget, ltOwner, x, y, ltMouseParams) {

}
```

weblayoutObject.onMouseIdle

```
weblayoutObject.onMouseIdle = function (ltTarget, ltOwner, x, y, ltMouseParams) {

}
```

weblayoutObject.onMouseLeave

```
weblayoutObject.onMouseLeave = function (ltTarget, ltOwner, x, y, ltMouseParams) {

}
```

weblayoutObject.onMouseMove

```
weblayoutObject.onMouseMove = function (ltTarget, ltOwner, x, y, ltMouseParams) {

}
```

weblayoutObject.onMouseTick

```
weblayoutObject.onMouseTick = function (ltTarget, ltOwner, x, y, ltMouseParams) {

}
```

weblayoutObject.onMouseUp

```
weblayoutObject.onMouseUp = function (ltTarget, ltOwner, x, y, ltMouseParams) {
```

```
}
```

weblayoutObject.onMouseWheel

```
weblayoutObject.onMouseWheel = function (ltTarget, ltOwner, x, y, ltMouseParams) {

}
```

weblayoutObject.onPopupDismissed

```
weblayoutObject.onPopupDismissed = function (ltTarget, ltOwner, reason, behaviorParams) {
    var ltPopupOwner = ..web.layout.element( behaviorParams.he )

}
```

weblayoutObject.onPopupDismissing

```
weblayoutObject.onPopupDismissing = function (ltTarget, ltOwner, reason, behaviorParams) {

}
```

weblayoutObject.onPopupReady

```
weblayoutObject.onPopupReady = function (ltTarget, ltOwner, reason, behaviorParams) {
    var ltPopupOwner = ..web.layout.element( behaviorParams.he )

}
```

weblayoutObject.onPopupRequest

```
weblayoutObject.onPopupRequest = function (ltTarget, ltOwner, reason, behaviorParams) {
    var ltPopupOwner = ..web.layout.element( behaviorParams.he )

}
```

weblayoutObject.onRequestTooltip

```
weblayoutObject.onRequestTooltip = function (ltTarget, ltOwner, reason, behaviorParams) {

}
```

weblayoutObject.onRowsDataRequest

```
weblayoutObject.onRowsDataRequest = function (ltTarget, ltOwner, reason, behaviorParams) {

}
```

weblayoutObject.onScrollEnd

```
weblayoutObject.onScrollEnd = function (ltTarget, ltOwner, pos, scrollParams) {

}
```

weblayoutObject.onScrollHome

```
weblayoutObject.onScrollHome = function (ltTarget, ltOwner, pos, scrollParams) {

}
```

weblayoutObject.onScrollPageMinus

```
weblayoutObject.onScrollPageMinus = function (ltTarget, ltOwner, pos, scrollParams) {

}
```

weblayoutObject.onScrollPagePlus

```
weblayoutObject.onScrollPagePlus = function (ltTarget, ltOwner, pos, scrollParams) {

}
```

weblayoutObject.onScrollPos

```
weblayoutObject.onScrollPos = function (ltTarget, ltOwner, pos, scrollParams) {

}
```

weblayoutObject.onScrollStepMinus

```
weblayoutObject.onScrollStepMinus = function (ltTarget, ltOwner, pos, scrollParams) {

}
```

weblayoutObject.onScrollStepPlus

```
weblayoutObject.onScrollStepPlus = function (ltTarget, ltOwner, pos, scrollParams) {

}
```

weblayoutObject.onSelectSelectionChanged

```
weblayoutObject.onSelectSelectionChanged = function (ltTarget, ltOwner, reason, behaviorParams) {
    var ltOption = ..web.layout.element( behaviorParams.he )

}
```

weblayoutObject.onSelectStateChanged

```
function (ltTarget, ltOwner, reason, behaviorParams) {
    var ltOption = ..web.layout.element( behaviorParams.he )
    —
}
```

weblayoutObject.onSize

```
weblayoutObject.onSize = function (ltOwner) {
```

```
}
```

weblayoutObject.onSliderRelease

```
weblayoutObject.onSliderRelease = function (ltTarget, ltOwner, pos,  scrollParams) {

}
```

weblayoutObject.onTableHeaderClick

```
weblayoutObject.onTableHeaderClick = function (ltTarget, ltOwner, cellIndex, behaviorParams) {

}
```

weblayoutObject.onTableRowClick

```
weblayoutObject.onTableRowClick = function (ltTarget, ltOwner, rowIndex, behaviorParams) {

}
```

weblayoutObject.onTableRowDblClick

```
weblayoutObject.onTableRowDblClick = function (ltTarget, ltOwner, rowIndex, behaviorParams) {

}
```

weblayoutObject.onTimer

```
weblayoutObject.onTimer = function (ltOwner, timerId) {

}
```

weblayoutObject.onUiStateChanged

```
weblayoutObject.onUiStateChanged = function (ltTarget, ltOwner, reason, behaviorParams) {

}
```

weblayoutObject.onVisualStateChanged

```
weblayoutObject.onVisualStateChanged = function (ltTarget, ltOwner, shown, behaviorParams) {

}
```

自动完成常量

_HL_ACTIVATE_CHILD=0x92
_HL_ALT_KEY_PRESSED=4
_HL_ANIMATION=0xA0
_HL_BUBBLING=0
_HL_BUTTON_CLICK=0
_HL_BUTTON_PRESS=1
_HL_BUTTON_STATE_CHANGED=2
_HL_BY_CODE=0

_HL_BY_DEL_CHAR=5
_HL_BY_DEL_CHARS=6
_HL_BY_INS_CHAR=3
_HL_BY_INS_CHARS=4
_HL_BY_KEY_CLICK=1
_HL_BY_KEY_NEXT=2
_HL_BY_KEY_PREV=3
_HL_BY_MOUSE=1
_HL_BY_MOUSE_CLICK=0
_HL_CLOSE_POPUP=0x9E
_HL_CONTEXT_MENU_REQUEST=0x10
_HL_CONTEXT_MENU_SETUP=0xF
_HL_CONTROL_KEY_PRESSED=1
_HL_CURSOR_APPSTARTING=0xB
_HL_CURSOR_ARROW=0
_HL_CURSOR_CROSS=3
_HL_CURSOR_DRAG_COPY=0xF
_HL_CURSOR_DRAG_MOVE=0xE
_HL_CURSOR_HAND=0xD
_HL_CURSOR_HELP=0xC
_HL_CURSOR_IBEAM=1
_HL_CURSOR_NO=0xA
_HL_CURSOR_SIZEALL=9
_HL_CURSOR_SIZENESW=6
_HL_CURSOR_SIZENS=8
_HL_CURSOR_SIZENWSE=5
_HL_CURSOR_SIZEWE=7
_HL_CURSOR_UPARROW=4
_HL_CURSOR_WAIT=2
_HL_DISABLED_STATUS_CHANGED=0x12
_HL_DOCUMENT_COMPLETE=0x98
_HL_DO_CLICK=0
_HL_DO_SWITCH_TAB=0x92
_HL_DRAGGING_COPY=2
_HL_DRAGGING_MOVE=1
_HL_DRAW_BACKGROUND=0
_HL_DRAW_CONTENT=1
_HL_DRAW_FOREGROUND=2
_HL_EDIT_VALUE_CHANGED=4
_HL_EDIT_VALUE_CHANGING=3
_HL_ELEMENT_COLLAPSED=0x90
_HL_ELEMENT_EXPANDED=0x91
_HL_EXC_COPY=1
_HL_EXC_LINK=4
_HL_EXC_MOVE=2
_HL_EXC_NONE=0
_HL_EXF_FILE=0x10
_HL_EXF_HTML=2
_HL_EXF_HYPERLINK=4
_HL_EXF_JSON=8
_HL_EXF_TEXT=1
_HL_EXF_UNDEFINED=0
_HL_FIRST_APPLICATION_EVENT_CODE=0x100
_HL_FIRST_APPLICATION_METHOD_ID=0x100
_HL_FORM_RESET=0x97
_HL_FORM_SUBMIT=0x96
_HL_GESTURE_FLAGS_ALL=0xFFFF
_HL_GESTURE_FLAG_PAN_HORIZONTAL=8
_HL_GESTURE_FLAG_PAN_VERTICAL=4
_HL_GESTURE_FLAG_PAN_WITH_GUTTER=0x4000
_HL_GESTURE_FLAG_PAN_WITH_INERTIA=0x8000
_HL_GESTURE_FLAG_ROTATE=2
_HL_GESTURE_FLAG_TAP1=0x10
_HL_GESTURE_FLAG_TAP2=0x20
_HL_GESTURE_FLAG_ZOOM=1
_HL_GESTURE_PAN=2
_HL_GESTURE_REQUEST=0
_HL_GESTURE_ROTATE=3
_HL_GESTURE_STATE_BEGIN=1
_HL_GESTURE_STATE_END=4
_HL_GESTURE_STATE_INERTIA=2
_HL_GESTURE_TAP1=4
_HL_GESTURE_TAP2=5
_HL_GESTURE_ZOOM=1
_HL_GET_TEXT_VALUE=1
_HL_GET_VALUE=0xFD
_HL_HANDLED=0x10000
_HL_HISTORY_DROP=0x9A
_HL_HISTORY_NEXT=0x9C
_HL_HISTORY_PRIOR=0x9B
_HL_HISTORY_PUSH=0x99
_HL_HISTORY_STATE_CHANGED=0x9D
_HL_HYPERLINK_CLICK=0x80
_HL_INIT_DATA_VIEW=0x93
_HL_IS_EMPTY=0xFC
_HL_MAIN_MOUSE_BUTTON=1
_HL_MENU_ITEM_ACTIVE=0xA
_HL_MENU_ITEM_CLICK=0xB
_HL_MIDDLE_MOUSE_BUTTON=4
_HL_NO_DRAGGING=0
_HL_POPUP_DISMISSED=9
_HL_POPUP_DISMISSING=0x13
_HL_POPUP_READY=8
_HL_POPUP_REQUEST=7

_HL_PROP_MOUSE_BUTTON=2
_HL_REQUEST_TOOLTIP=0x9F
_HL_ROWS_DATA_REQUEST=0x94
_HL_SCROLL_BAR_GET_VALUE=6
_HL_SCROLL_BAR_SET_VALUE=7
_HL_SELECT_SELECTION_CHANGED=5
_HL_SELECT_STATE_CHANGED=6
_HL_SET_TEXT_VALUE=2
_HL_SET_VALUE=0xFE
_HL_SHIFT_KEY_PRESSED=2
_HL_SINKING=0x8000
_HL_SYNTHESIZED=2
_HL_TABLE_HEADER_CLICK=0x81
_HL_TABLE_ROW_CLICK=0x82
_HL_TABLE_ROW_DBL_CLICK=0x83
_HL_TEXT_EDIT_CHAR_POS_AT_XY=0xB
_HL_TEXT_EDIT_GET_CARET_POSITION=8
_HL_TEXT_EDIT_GET_SELECTION=3
_HL_TEXT_EDIT_GET_SELECTION_HTML=0xA
_HL_TEXT_EDIT_GET_SELECTION_TEXT=9
_HL_TEXT_EDIT_REPLACE_SELECTION=5
_HL_TEXT_EDIT_SET_SELECTION=4
_HL_UI_STATE_CHANGED=0x95
_HL_VISIUAL_STATUS_CHANGED=0x11
_HL_X1_MOUSE_BUTTON=8
_HL_X2_MOUSE_BUTTON=0x10
_HL_XCALL=0xFF
_HL_X_DRAG=2
_HL_X_DRAG_ENTER=0
_HL_X_DRAG_LEAVE=1
_HL_X_DROP=3