

特征	说明	示例
命名空间	引用模型元素时, 您必须对这些元素添加前缀。存在下列前缀: <b>dp:</b> (表示数据源项), <b>ev:</b> (表示事件), <b>v:</b> (表示局部变量), <b>f:</b> (表示函数)	<pre>dp:x = 100; // set a datapool item fire ev:back(); // fire an event f:trace_string("hello world"); // call a function</pre>
访问数据源项	通过将数据源项放置在赋值左侧来编写此项。通过在表达式中的任何其他位置使用数据源项来读取此项。重定向引用  (=>) 是数据源项赋值的特殊形式。	<pre>dp:x = 5; // writing to x dp:x = dp:y + dp:z; // reading y and z length dp:aList; // read the length of a list datapool item dp:refX =&gt; dp:x; // redirect link</pre>
发送事件	语法: <b>fire ev:</b> <identifier>{<parameter-list>;}  可以在超时后发送事件。 可以使用 <b>cancel_fire</b> 表达式取消此延迟事件。  语法: <b>fire_delayed</b> <timeout>, <b>ev:</b> <identifier>{<parameter-list>;} <b>cancel_fire ev:</b> <identifier>;	<pre>fire ev:back(); fire ev:mouseClick(10, 20);  fire_delayed 3000, ev:back(); // send the event "back"                                in 3 seconds. cancel_fire ev:back; // cancel the event</pre>
对事件做出响应	要对事件做出响应, 请使用 <b>match_event</b> 。这是 <b>if-then-else</b> 语句的特殊形式。 <b>if</b> 和 <b>else</b> 分支必须始终具有相同类型。如果在赋值右侧使用, 则 <b>else</b> 分支是必需的。  语法: <b>match_event v:</b> <identifier> = <b>ev:</b> <identifier> <b>in</b> <sequence> <b>else</b> <sequence>	<pre>match_event v:event = ev:back in {     f:trace_string("back event received"); }  v:this.x = match_event v:event = ev:back in 10 else 0;</pre>
访问事件参数	<b>match_event</b> 的 <b>in</b> 表达式可以访问事件参数。使用点标记访问事件参数。	<pre>match_event v:event = ev:mouseClick in {     v:this.x = v:event.x;     v:this.y = v:event.y; }</pre>
访问控件属性	如果某个脚本是小组件(小组件操作、输入响应)的组成部分, 则它可以访问该小组件的属性。通过引用当前小组件的变量, 名为 <b>v:this</b> 的特殊局部变量可用。使用点标记对控件属性进行寻址。	<pre>v:this.text = "hello world"; v:this.x = 10;</pre>
浏览小组件树	如果某个脚本是小组件的组成部分, 则它可以访问其他小组件的属性。使用控件树导航运算符:->。要访问父代小组件, 请使用标识符:^。	<pre>v:this-&gt;^-&gt;caption.text = "Play"; // goto parent, goto caption, property text v:this-&gt;^..x = 1; // goto parent, property x</pre>
字符串格式设置	运算符用于连接字符串。有关更多字符串转换函数, 请参阅文档。	<pre>v:this.text = "current speed: " + f:int2string(dp:speed) + "km/h";</pre>
字符串比较	要比较两个字符串(区分大小写), 请使用相等运算符 <b>==</b> 或 <b>!=</b> 。 要比较两个字符串(不区分大小写), 请使用相等运算符 <b>=Aa=</b> 。	<pre>"name" == "NAME" // false "name" != "NAME" // true "name" =Aa= "NAME" // true</pre>
更改语言	要更改 EB GUIDE 模型中所有数据池项的语言, 请使用 <b>setLanguage</b> 。该操作以异步方式执行。 语法: <b>f:setLanguage</b> (l:<identifier>, bool<isCoreScope>)	<pre>f:setLanguage(l:Standard, false) // changes language to the standard language at the model scope f:setLanguage(l:German, true) // changes language to German at the core scope</pre>

特征	描述	示例
更改外观	要更改 EB GUIDE 模型中所有数据池项的外观, 请使用 <code>setSkin</code> 。该操作以异步方式执行。 语法: <code>f:setSkin(s:&lt;identifier&gt;, bool&lt;isCoreScope&gt;)</code>	<pre>f:setSkin(s:Standard, true) // changes to the standard skin                              at the core scope f:setSkin(s:"myskin", false) // changes to a user-defined skin                              at the model scope</pre>
常量	编写的字符串常量可以不带引号。 颜色常量是 RGBA 四重。	<pre>"hello world" // string constant Napoleon // string constant 5 // integer constant color:0,235,0,255 // EB green</pre>
算术、逻辑和赋值运算符	加法和字符串连接: +, 减法: -, 乘法: *, 除法: /, 取模: %, 大于: >, 小于: <, 大于或等于: >=, 小于或等于: <=, 等于: ==, 不等于: !=, 和: &&, 或:   , 非: !, 赋值: =, 赋值增加: +=, 赋值减少: -=	<pre>dp.myString = "Hello" + "World"; dp.count += 1; // increment one</pre>
序列	序列可以是单个表达式, 也可以括在大括号中的一系列表达式。序列中的最后一个表达式为序列的值。	<pre>if( dp:something )     dp:x = 5; // single expression if( dp:other ) {     dp:x = 5; // sequence enclosed     dp:y = 10; // in curly braces }</pre>
局部变量	使用 <code>let</code> 绑定引入局部变量。不允许使用未初始化的变量。 <code>let</code> 绑定可以进行嵌套。 语法: <code>let v:&lt;identifier&gt; = &lt;expression&gt;;     v:&lt;identifier2&gt; = &lt;expression&gt;;     ...     in &lt;sequence&gt;</code>	<pre>let v:x = 42;     v:text = "hello world"; in {     v:this.x = v:x;     v:this.text = v:text; }</pre>
while 循环	<b>while</b> 循环包含两个表达式: 条件表达式和主体表达式。在条件生成 <code>false</code> 之前, 将对主体重复求值。 语法: <code>while( &lt;expression&gt; ) &lt;sequence&gt;</code>	<pre>dp:i = 0; while( dp:i &lt;= 10 ) {     dp:sum += i;     dp:i += 1; }</pre>
If-then-else	<b>if-then-else</b> 的行为类似于 C 和 Java 中的三元条件运算符。如果在赋值右侧使用它, 则 <b>else</b> 分支是必需的并且这两个分支必须具有相同类型。 语法: <code>if( &lt;expression&gt; ) &lt;sequence&gt; else &lt;sequence&gt;</code>	<pre>if( dp:buttonClicked ) {     v:this.x = dp:x; } else {     v:this.x = 0; }  v:this.x = if( dp:buttonClicked ) dp:x else 0;</pre>
注释	允许 C 样式块注释和 C++ 样式行注释。	<pre>/* this is a C style block comment */ // this is a C++ style line comment</pre>
返回值	脚本中的最后一个表达式是返回值。 要强制使用无效类型的返回值, 请使用 <code>unit</code> 或 <code>{}</code>	<pre>dp:x + 2; // returns datapool item x plus 2</pre>