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：
	
	dp:(表示数据源项)，**ev:**(表示事件)，**v:**(表示局部变量)，**f:**(表示函数)
		dp:x = 100; // set a datapool item
		fire ev:back(); // fire an event
		f:trace_string("hello world"); // call a function

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。
		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 => dp:x; // redirect link

发送事件	语法:
	fire ev:<identifier>**(**<parameter-list>**);**
	
	可以在超时后发送事件。
	可以使用 **cancel_fire**表达式取消此延迟事件。
	
	语法:
	fire_delayed <timeout>, **ev:**<identifier>**(**<parameter-list>**);**
	cancel_fire ev:<identifier>**;**
		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

对事件做出响应	要对事件做出响应，请使用**match_event**。这是**if-then-else**语句的特殊形式。**If** 和 **else**分支必须始终具有相同类型。如果在赋值右侧使用，则 **else**分支是必需的。
	
	语法:
	match_event v:<identifier> **= ev:**<identifier>
	in <sequence>
	else <sequence>
		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;

访问事件参数	**match_event**的**in** 表达式可以访问事件参数。使用点标记访问事件参数。	match_event v:event = ev:mouseClick in {
		v:this.x = v:event.x;
		v:this.y = v:event.y;
		}

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 **v:this**的特殊局部变量可用。使用点标记对控件属性进行寻址。	v:this.text = "hello world"; v:this.x = 10;

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: ^.	v:this->^->caption.text = "Play"; // goto parent, goto caption, property text
		v:this->^.x-= 1; // goto parent, property x

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。	v:this.text = "current speed: " + f:int2string(dp:speed) + "km/h";
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 **== or !=**。 要比较两个字符串(不区分大小写)，请使用相等运算符 **=Aa=**。	"name" == "NAME" // false "name" != "NAME" // true "name" =Aa= "NAME" // true
更改语言	要更改 EB GUIDE 模型中所有数据源项的语言，请使用**language**。此操作是异步执行。 语法: **f:language(l:**<identifier>**)**	f:language(l:Standard) // changes language to the standard language f:language(l:German) // changes language to German

特征	描述	示例
更改外观	要更改 EB GUIDE 模型中所有数据源项的外观，请使用**skin**。此操作是异步执行。 语法: **f:skin(s:**<identifier>**)**	f:skin(s:Standard) // changes to the standard skin f:skin(s:“myskin”) // changes to a user-defined skin
常量	编写的字符串常量可以不带引号。 颜色常量是 RGBA 四重。	“hello world” // string constant Napoleon // string constant 5 // integer constant color:0,235,0,255 // EB green
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: **-=**	dp:myString = “Hello” + “World”; dp:count += 1; // increment one

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。	if(dp:something)
		  dp:x = 5; // single expression
		if(dp:other) {
		  dp:x = 5; // sequence enclosed
		  dp:y = 10; // in curly braces
		}

局部变量	使用let绑定引入局部变量。不允许使用未初始化的变量。
	let绑定可以进行嵌套。
	
	语法:
	let v:<identifier> **=** <expression>**;**
	  v:<identifier2> **=** <expression>**;**
	  ...
	  in <sequence>
		let v:x = 42;
		  v:text = “hello world”;
		in {
		  v:this.x = v:x;
		  v:this.text = v:text;
		}

while 循环	**while**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 false 之前，将对主体重复求值。
	
	语法:
	while(<expression> **)** <sequence>
		dp:i = 0;
		while(dp:i <= 10) {
		  dp:sum += i;
		  dp:i += 1;
		}

If-then-else	**If-then-else** 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 **else**分支是必需的并且这两个分支必须具有相同类型。
	
	语法:
	if(<expression> **)** <sequence> **else** <sequence>
		if(dp:buttonClicked) {
		  v:this.x = dp:x;
		}
		else {
		  v:this.x = 0;
		}
		
		v:this.x = if(dp:buttonClicked) dp:x else 0;

注释	允许 C 样式块注释和 C++ 样式行注释。	/* this is a C style block comment */ // this is a C++ style line comment
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 **unit**或 **{}**	dp:x + 2; // returns datapool item x plus 2