Runtime Functions

Table of Contents

CreateObject(classname as String, [optional parameters]) as Object

Creates a BrightScript Component of class *classname* specified. Return invalid if the object creation fails. Some Objects have optional parameters in their constructor that are passed after *name*.

Example

```
app_mgr = CreateObject("roAppManager")
section = CreateObject("roRegistrySection", "Data")
```

Type(variable, [optional version]) as String

Returns the type of a variable and/or object. See the BrightScript Component specification for a list of types.

Example

```
Print type(5) 'returns a 2.1 compatible type
Print type("my string", 3) 'return a BrightScript 3 type
```

GetGlobalAA() as Object

Each script has a global Associative Array. It can be fetched with this function.
New in BrightScript 3.

Box(x as Dynamic) as Object

Box() will return an object version of an intrinsic type, or pass through an object if given one.

```
bo = box("string")
bo = box(bo) ' no change to bo
```

Run(filename as String [, Args…]) As dynamic

Run(filenamearray as Object [, Args…]) As dynamic

The Run function can be used to compile and run a script dynamically.
If a string is passed in the filename parameter, the file specified by that path is compiled and run.
If an array of strings is passed in the filename parameter, then all files specified are compiled together, then run.

Arguments may be passed to the script's Main function, and that script may return a result value.

Example

```
Sub Main()
    Run("pkg:/test.brs")
    BreakIfRunError(LINE_NUM)
    Print Run("test2.brs", "arg 1", "arg 2")
    if Run(["pkg:/file1.brs","pkg:/file2.brs"])<>4 then stop
    BreakIfRunError(LINE_NUM)     stop
End Sub


Sub BreakIfRunError(ln)
    el=GetLastRunCompileError()
    if el=invalid then
        el=GetLastRunRuntimeError()
        if el=&hFC or el=&hE2 then return
        'FC==ERR_NORMAL_END, E2=ERR_VALUE_RETURN
        print "Runtime Error (line ";ln;"): ";el
        stop
    else
        print "compile error (line ";ln;")"
        for each e in el
            for each i in e
                print i;": ";e[i]
            end for
        end for
        stop
   end if
End Sub
```

Eval(code as String) as Dynamic

> Eval() is deprecated effective immediately. Use of Eval() will cause compilation or runtime errors, see Roku Channel Manifest.
>
> Use roXMLElement.parse() or parseJSON() instead of Eval() when initializing data.

Eval can be used to run a code snippet in the context of the current function. It performs a compile, and then the bytecode execution.

If a compilation error occurs, no bytecode execution is performed, and Eval returns an roList with one or more compile errors. Each list entry is an roAssociativeArray with ERRNO and ERRSTR keys describing the error.

If compilation succeeds, bytecode execution is performed and the integer runtime error code is returned. These are the same error codes as returned by GetLastRunRuntimeError().

Eval() can be usefully in two cases. The first is when you need to dynamically generate code at runtime.

The other is if you need to execute a statement that could result in a runtime error, but you don't want code execution to stop.

Example:

```
Print Eval("n=1/0")
```

Outputs:

```
20
```

(Note that 20 = &h14 = ERR_DIV_ZERO = Divide by Zero error).

GetLastRunCompileError() as Object

Returns an roList of compile errors, or invalid if no errors. Each list entry is an roAssociativeArray with the keys: ERRNO, ERRSTR, FILESPEC, and LINENO.

GetLastRunRuntimeError() as Integer

Returns an error code result after the last script Run().
These are normal:
&hFF==ERR_OKAY
&hFC==ERR_NORMAL_END
&hE2==ERR_VALUE_RETURN

Example: Assign variables to common runtime errors

```
ERR_USE_OF_UNINIT_VAR = &hE9
ERR_DIV_ZERO = &h14
ERR_TM = &h18
ERR_USE_OF_UNINIT_VAR = &hE9
ERR_RO2 = &hF4
ERR_RO4 = &hEC
ERR_SYNTAX = 2
ERR_WRONG_NUM_PARAM = &hF1
```