Basic data types

22 сентября 2015 г.
10:29

```
// Integer types
//

// unsigned
esU8
esU16
esU32
esU64

// signed
esI8
esI16
esI32
esI64

// 64-bit Integer date time, as ms since 2000-01-01
esDT

// 32 bit float
esF

// 64 bit float (double)
esD
```

Operators

18 ноября 2011 г.
12:57

// generic
Pre and post increment ++
Pre and post increment --
Logical or ||
Logical and &&
Logical not !
Bitwise and &
Bitwise or |
Bitwise xor ^
Bitwise not ~
Left shift <<
Right shift >>
Addition +
Subtraction -
Multiplication *
Division /
Modulus %

// comparison
Less <
Greater >
Equality ==
Not equal !=
Less or equal <=
Greater or equal >=

// membership| implicit equality check
in

// assignments
Assignment =
Assignment with decrement -=
Assignment with increment +=
Assignment with division /=
Assignment with modulus %=
Assignment multiplication *=
Assignment with bitwise or |=
Assignment with bitwise and &=
Assignment with bitwise xor ^=
Assignment with left shift <<=
Assignment with right shift >>=

// access operators
Object method access .
Object field access .
Object variable access .
Object property access $
Object/instance attribute access @
Scope specifier ::
Variant service explicit access #
Enumeration value access $$

Reserved words

18 ноября 2011 г.
12:57

Require
extern
var
const
new
delete
function
return
object
extends
enum
true
false
if
in
else
for
foreach
while
do
break
continue
switch
case
default
label
goto

Variables, constants

18 ноября 2011 г.
12:56

```
// variable declaration
var v1_0[ = value][, v1_1[= value], … , v1_n];
var v2_0[ = value][, v2_1[= value], … , v2_n];

// array variables
//
// empty array
var a = [];
// initialized array
var a = [1, 2, 3, "some item"];

// constant declaration
const c1 = immediate_value;
```

Strings

22 сентября 2015 г.
20:59

```
// Unicode (wide) strings
var str = "International wide string";

// Byte strings
var bstr = B"Byte string";
```

Collections, arrays

22 сентября 2015 г.
21:03

```
var collection0 = [];  // An empty collection

// Append elements to collection
collection0 += 1;
collection0 += 324;
collection0 += "Some string";

EsScriptDebug::log("Collection item[%d]=%s", 2, collection0[2]);

// Reset collection
collection  =  [];

var collection1 = [1, 23, "SomeString", SomeEnumeration$$enumerationMember ]; // Collection
initializer

// Collection iteration
var item;
foreach(item in collection1)
        EsScriptDebug::log( item );
```

Attributes

18 февраля 2012 г.
12:05

// attributes declaration
[@attribute1_name=attribute1_value;
..
@attributeN_name=attributeN_value;]

Known attributes, **i**=instance (e.g. fields-specific), **c**=class:
@help $_{i, c}$ = string for runtime help generators
@label $_{i}$ = string for labeling related GUI elements
@default $_{i}$ = value for field (re)initialization
@restriction $_{i}$ = value for simple validation checking
@fixedSize $_{c}$ = value, specifying that fixed size in bytes

Enumerations

2 декабря 2011 г.
22:27

Declaration

enum EnumerationName
[attribute declaration]
{
	value0name=value [, "value label string"];
	...
	valueNname=value [, "value label string"];
}

Value access

<EnumerationName>$$<value name>

Label access

<EnumerationName>$$<value name>$$label

All value names

<EnumerationName>$symbols

All values access

<EnumerationName>$values

All labels access

<EnumerationName>$labels

Functions

18 ноября 2011 г.
14:15

```
// function declaration
function <function name>([p0, p1 .. pn])
[attributes declaration]
[variable declaration]
{
        [return [value];]
}

// function call
var v = f();
```

Script objects

18 ноября 2011 г.
14:26

```
// Object declaration
object BaseObj
[attributes declaration]
{
	// variable declarations
	var v0, … vn;

	// field declarations
	[POD type]or[ScriptObject] <fieldName>;

	esU8 cnt;
	[field attributes declaration]

	// member function declarations

	// property declarations
	property <property name>;
	[property attributes]
	read: { return <field|variable|property accessor function>; } //< Optional read specifier
	write: { <field|variable> = __value; or <property setter function>(__value); } //< Optional write
	specifier
}

// extended class declaration
object ExtObj extends BaseObj
[attributes declaration]
{
	esU8 ua8_0[cnt];
	esF f;
	esU16 ua16_0[6-cnt];
	if( cnt > 4 )
	{
		esF cf0;
		esI16 i0;
		esU16 ua16_1[cnt];
		esI8 i1;
	}
	else
	{
		esF cf1;
		if( 2 == cnt )
		{
			esU32 ttt;
			esF cf0_1;
		}
		else
			esU8 ua8_1[10-cnt];
	}
	esU32 ui;

	new( p0 )
```

```
	{
		f = p0;
	}
}
```

Outputting script debug trace in Eco-E scripting console

22 сентября 2015 г.
20:48

```
// Simple message trace
EsScriptDebug::log("message");

// Formatted message trace
EsScriptDebug::log(<fmt string>, param0,...paramN);
```

Calling functionality reflected from C++ libraries or script objects

22 сентября 2015 г.
20:51

```
// Reflected static methods
<Reflected object name>::<method call>;

var chnl = EsChannelIoFactory::channelCreate("EsChannelIoUart");

// Calling member functions
<object>.<method call>;
If( chnl.open() )
        chnl.bytesPut(B"TestByteString", 14);
```

Sample code for script debugging implemented in script

22 сентября 2015 г.
21:11

```
// Additional script debug functionality
//
function debugValueDump(val)
var idx, result = "";
{
 if( !val#isEmpty() )
 {
  if( EsVariantType$$VAR_STRING == val#typeGet() ) // string
   result = val + "\n";
  else if( val.isIndexed() )
  {
   result = "indexed contents...\n";
   for(idx = 0; idx < val.countGet(); ++idx)
    result += EsStr::format("[%d]=%s", idx, debugValueDump( val.itemGet(idx) ));
  }
  else if( val#isObject() )
  {
   // object type, size and offset
   result += EsStr::format("type '%s', size=%d, offset=%d\n", val$type, val$size, val$offset);
   if( val.hasProperty("value") )
    result += "value=" + debugValueDump(val$value);
  }
  else
   result = val#asString() + "\n";
 }
 else
  result = "null\n";

 return result;
}

// helper functions for object printout
function debugObjectDump(obj)
var names, tmp, idx, result;
{
 // object value dump
 result = debugValueDump(obj);
 // object attributes
 names = obj$attributeNames;
 for(idx = 0; idx < names.countGet(); ++idx)
 {
  tmp = obj.attributeGet(names[idx]);
  result += EsStr::format("attribute '%s'=%s\n", names[idx], tmp);
 }
 // object fields
 names = obj$fieldNames;
 for(idx = 0; idx < names.countGet(); ++idx)
 {
  tmp = obj.fieldGet(names[idx]);
  result += EsStr::format("field '%s::%s' dump\n", obj$type, names[idx]);
  result += debugObjectDump(tmp);
```

```
  }

  return result;
}
```

Reflected Class Declaration and Definition

21 февраля 2012 г.
7:40

1. Stand-alone or base reflected class, full refcounted implementation

a. Declaration

```
class ClassName : Public EsReflectedClassIntf
{
...
// declare required reflected class interface entries
ES_DECL_REFLECTED_CLASS_BASE( ClassName )
// implement interface table
// interface support map
//
ES_INTF_MAP_BEGIN( ClassName  )
        ES_INTF_SUPPORTS( ClassName , EsReflectedClassIntf )
ES_INTF_MAP_END
{
        // destroy if refcount reaches 0
        delete this;
}
…
};
```

b. Implementation

```
ES_DECL_BASE_CLASS_INFO_BEGIN(ClassName, wxT("optional description string or NO_CLASS_DESCR"))
        // reflected properties and methods  entries go here
ES_DECL_CLASS_INFO_END
```