EScript

Short Presentation of a Scripting Language

Benjamin Eikel

May 4, 2012

Overview

Overview

What is EScript?

EScript . . .

- is an object-oriented scripting language.
- is compiled and executed by a virtual machine.
- has a similar syntax to C.
- was developed to use C++ objects from scripts easily.

What is EScript?

EScript . . .

- is released under a free software license.
- is available from `http://escript.berlios.de/`.
- can be built using CMake.
- has a command-line interpreter.
- can be used internally by other C++ projects (e.g. PADrend).

First Example

- EScript files should have the extension `.escript`.
- The EScript parser analyzes the script file line by line.
- A simple script:

```
out("Hello, world!\n");
```

Overview

Simple Types

Number

```
1        27.4        0x1a        25 / 5        3 + 4
```

String

```
"an"        'example'        "hello"+ ', ' + "world"
```

Bool

true **false**

Void

void

Type Conversion

No conversion to false

```
outln(false || false); // Output: false
outln(false || 0); // Output: true
outln(false || ""); // Output: true
```

Type Conversion

No conversion to false

```
outln(false || false); // Output: false
outln(false || 0); // Output: true
outln(false || ""); // Output: true
```

Conversion of String to Number

```
outln((60 + "4").sqrt()); // Output: 8
outln((10 * "10").log(10)); // Output: 2
```

Type Conversion

No conversion to false

```
outln(false || false); // Output: false
outln(false || 0); // Output: true
outln(false || ""); // Output: true
```

Conversion of String to Number

```
outln((60 + "4").sqrt()); // Output: 8
outln((10 * "10").log(10)); // Output: 2
```

Conversion of Number to String

```
outln("4" + 60); // Output: 460
outln("12" + 3); // Output: 123
```

Variables, Comments

var

```
/*
    Declaring a variable is done using the keyword
    "var", an identifier, an equation sign, and an
    expression on the right side. The type of the
    variable is deduced from the expression on the
    right side.
 */
var xPos = 500 - 80 / 2;

// The variable "message" will be of type String
var message = "Please click the button";

// Dynamically change the type to Number
message = 5;
```

Functions

fn

```
var square = fn(num) {
    return num * num;
};
var a = square(5);
var b = square(4.2);
```

Advanced Types (1)

Array

```
var numbers = [3, 23, 7, 3, 100, 1, 35];
var colors = ["red", "green", "blue"];
```

Map

```
var fruits = {
    "lemon" : "yellow",
    "cherry" : "red"
};
fruits["apple"] = "green";
```

Advanced Types (2)

ExtObject

```
var car = new ExtObject();
car.color := "red";
car.speed := 190;
car.outputDesc := fn() {
    out("This is a ", this.color, " car ");
    out("with top speed ", this.speed, ".\n");
};

...

car.speed = 185;
car.outputDesc();
```

Output: This is a red car with top speed 185.

Advanced Types (3)

Type

```
var Shape = new Type();
Shape.color := "white";

// New type that is derived from Shape
var Polygon = new Type(Shape);
Polygon.numVertices := 3;

// New type that is derived from Shape
var Circle = new Type(Shape);
Circle.radius := 0;

var circle = new Circle();
circle.color = "red";
circle.radius = 5;
```

Overview

Conditionals (1)

if

```
var result = /* some function */;
if(result) {
    out("Success");
} else {
    out("Failure");
}

var num = /* some number */;
if(num < 0) {
    out("Too small");
} else if(num >= 0 && num <= 100) {
    out("Range okay");
} else {
    out("Too large");
}
```

Conditionals (2)

? (conditional operator)

```
var num = /* some number */;
var positive = (num > 0) ? true : false;
```

Conditionals (2)

? (conditional operator)

```
var num = /* some number */;
var positive = (num > 0) ? true : false;
```

Note: There is no `switch` in EScript.

Loops (1)

while

```
var tasks = [/* some tasks */];
while(!tasks.empty()) {
    var firstTask = tasks.front();
    tasks.popFront();
    // do something with first task
}
```

Loops (2)

for

```
var sum = 0;
for(var i = 0; i < 100; ++i) {
    sum += i;
}
out("Sum of numbers: ", sum, "\n");
```

Loops (3)

foreach

```
var chars = ["a", "c", "k", "b", "d", "x", "j"];
foreach(chars as var i, var c) {
    if(c == "x") {
        out("Character \"x\" found at index " + i);
        break;
    }
}
```

Output: Character "x" found at index 5

Overview

Delegation

Call a function on another object.

Example

```
var printOut = fn() {
    out("I am a " + this.color + " node.\n");
};

var nodeRed = new ExtObject();
nodeRed.color := "red";
var nodeBlack = new ExtObject();
nodeBlack.color := "black";

var printOutRed = nodeRed -> printOut;
var printOutBlack = nodeBlack -> printOut;

printOutRed(); // Output: I am a red node.
printOutBlack(); // Output: I am a black node.
```

Properties

Example

```
var Polygon = new Type();
Polygon.vertices @(private, init) := Array;
Polygon.shapeType @(const) := "Polygon";

Polygon.getNumVertices := fn() {
    return this.vertices.count();
};

var polygon = new Polygon();
polygon.getNumVertices();
```

Overview

Factorial

Factorial: $n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$ $0! = 1$

Example

```
var factorialRecursive = fn(Number n) {
    return (n == 0) ? 1 : thisFn(n - 1) * n;
};
var factorialIterative = fn(Number n) {
    var product = 1;
    for(var i = 2; i <= n; ++i) {
        product *= i;
    }
    return product;
};

outln(factorialRecursive(6)); // Output: 720
outln(factorialIterative(7)); // Output: 5040
```

Player

Example

```
var Player = new Type();
Player.x @(private) := 0;
Player.y @(private) := 0;
Player.move ::= fn(Number dx, Number dy) {
    this.x += dx;
    this.y += dy;
};
Player.printPos ::= fn() {
    outln("Position: (", this.x, ", ", this.y, ")");
};

var playerA = new Player();
playerA.move(5, 7);
playerA.printPos(); // Output: Position: (5, 7)
```

Further Documentation

You can find additional documentation in `EScript/docs/Introduction.html`.