EScript

Kurzvorstellung einer Skriptsprache

Benjamin Eikel

12. Februar 2013

Übersicht

Übersicht

Was ist EScript?

EScript . . .

- ist eine objektorientierte Skriptsprache.
- wird übersetzt und zur Laufzeit durch eine virtuelle Maschine ausgeführt.
- hat eine ähnliche Syntax wie C.
- wurde entwickelt, um C++-Objekte einfach in Skripten verwenden zu können.

Was ist EScript?

EScript . . .

- ist unter einer freien Softwarelizenz veröffentlicht.
- ist erhältlich unter `http://escript.berlios.de/`.
- kann mit CMake gebaut werden.
- hat einen Kommandozeileninterpretierer.
- kann intern von anderen C++-Projekten benutzt werden (z. B. PADrend).

Erstes Beispiel

- EScript-Dateien sollten die Endung `.escript` haben.
- Der EScript-Parser analysiert das Skript Zeile für Zeile.
- Ein einfaches Skript:

```
out("Hallo Welt!\n");
```

Übersicht

Einfache Typen

Number

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

String

```
"ein"        'beispiel'        "hallo"+ "welt"
```

Bool

true **false**

Void

void

Typkonvertierung

Keine Konvertierung zu false

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

Typkonvertierung

Keine Konvertierung zu false

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

Konvertierung von String nach Number

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

Typkonvertierung

Keine Konvertierung zu false

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

Konvertierung von String nach Number

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

Konvertierung von Number nach String

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

Variablen, Kommentare

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;
```

Funktionen

fn

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

Komplexere Typen (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";
```

Komplexere Typen (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();
```

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

Komplexere Typen (3)

Type

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

// Neuer Typ, der von Shape erbt
var Polygon = new Type(Shape);
Polygon.numVertices := 3;

// Neuer Typ, der von Shape erbt
var Circle = new Type(Shape);
Circle.radius := 0;

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

Übersicht

Abfragen (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");
}
```

Abfragen (2)

? (conditional operator)

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

Abfragen (2)

? (conditional operator)

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

Hinweis: Es gibt kein `switch` in EScript.

Schleifen (1)

while

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

Schleifen (2)

for

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

Schleifen (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;
    }
}
```

Ausgabe: Character "x" found at index 5

Übersicht

Delegation

Aufruf einer Funktion auf einem anderen Objekt.

Beispiel

```
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.
```

Attribute

„Attribut“ ist nicht das korrekte Wort

Beispiel

```
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();
```

Übersicht

Fakultät

Rekursion (thisFn) vorher einführen

Typüberprüfung für Parameter vorher einführen

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

Beispiel

```
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
```

Spieler

::= vorher einführen

Beispiel

```
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)
```

Zusätzliche Dokumentation

Zusätzliche Dokumentation befindet sich in `EScript/docs/Introduction.html`.