EScript

Kurzvorstellung einer Skriptsprache

Benjamin Eikel

13. Februar 2012

Übersicht

Was ist EScript?

EScript . . .

- ist eine interpretierte, objektorientierte Skriptsprache.
- hat eine ähnliche Syntax wie C.
- wurde entwickelt, um C++-Objekte einfach in Skripten verwenden zu können.
- ist unter einer freien Softwarelizenz veröffentlicht.
- ist erhältlich unter `http://escript.berlios.de/`.

Erste Beispiele

- EScript-Dateien sollten die Endung `.escript` haben.
- Ein einfaches Skript:

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

Einfache Typen

Number Beispiele: `1`, `27.4`, `0x1a`, `25 / 5`, `3 + 4`

String Beispiele: `"ein"`, `'beispiel'`, `"hallo"+ "welt"`

Bool **`true`** oder **`false`**

Void **`void`**

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();
```

Output: 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;
```

Abfragen

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");
}
```

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 c) {
    if(c == "x") {
        out("Character \"x\" found.");
        break;
    }
}
```

Fakultät

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

Implementierung

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

Spieler

Implementierung

```
var Player = new Type();
Player.x := 0;
Player.y := 0;
var movePlayer = fn(player, Number dx, Number dy) {
    player.x += dx;
    player.y += dy;
};
var printPos = fn(player) {
    out("Player position: (", player.x);
    out(", ", player.y, ")\n");
};

var playerA = new Player();
movePlayer(playerA, 5, 7);
printPos(playerA);
```