



- 1 Einführung
- 2 Datentypen
- 3 Kontrollstrukturen
- 4 Weitere Funktionalität
- 5 Beispiele



- 1 Einführung
- 2 Datentypen
- 3 Kontrollstrukturen
- 4 Weitere Funktionalität
- 5 Beispiele



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.



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).

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

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



1 Einführung

2 Datentypen

3 Kontrollstrukturen

4 Weitere Funktionalität

5 Beispiele

Number

1

27.4

0x1a

25 / 5

3 + 4

String

"ein"

'beispiel'

"hallo" + "welt"

Bool

true

false

Void

void

Keine Konvertierung zu false

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

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
```

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
```

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;
```

fn

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

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";
```

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.

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;
```




- 1 Einführung
- 2 Datentypen
- 3 Kontrollstrukturen**
- 4 Weitere Funktionalität
- 5 Beispiele

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");
}
```



? (conditional operator)

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



? (conditional operator)

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

Hinweis: Es gibt kein `switch` in EScript.

while

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

for

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

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



- 1 Einführung
- 2 Datentypen
- 3 Kontrollstrukturen
- 4 Weitere Funktionalität**
- 5 Beispiele

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.
```

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();
```



- 1 Einführung
- 2 Datentypen
- 3 Kontrollstrukturen
- 4 Weitere Funktionalität
- 5 Beispiele**

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

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
```

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



Vielen Dank für Ihre Aufmerksamkeit!

Benjamin Eikel

Heinz Nixdorf Institut
& Institut für Informatik
Universität Paderborn

Adresse: Fürstenallee 11
33102 Paderborn
Deutschland

Telefon: +49 5251 60-6452
Fax: +49 5251 60-6482
E-Mail: eikel@upb.de
Web: <http://www.hni.upb.de/en/alg/>

Benjamin Eikel ist Stipendiat der
 **International Graduate School**
Dynamic Intelligent Systems

