Catscript Guide

Introduction

Catscript is a small statically typed scripting language that compiles to JVM bytecode featuring both evaluation and compilation. It is designed to take the best features of Java and combine it with features inspired from other languages to create an ideal language to work with.

Here are some features:

Features

###Comments Comments are used in Catscript using the "//" for single line comments and "/* */" for block comments

```
/*
*This is
a multiline comment
*/
var x = 8
//This is a single line comment
print(x)
```

###Types Catscript supports basic types for variables. These types can be both explicitly declared and inferred. Catscript also shares namespaces, variable names from all fields and sub-scopes are stored in the same location. Here is the small list of Catscript's type system:

- int - a 32 bit integer
- null - a null type
- bool - boolean value
- string - Java string object
- list - a list of objects
- object - any value

For loops

Using the 'in' keyword like most other languages, for loops in Catscript are used to iterate through contents in an array as shown here:

```
var list = ["a", "b", "c", "d"]
for(i in list){
print(i)
}
```

This returns "abcd". i is assigned to each character within the array and printed as the loop iterates through the array.

If statements

If statements in Catscript identical to Java. Using the if, else if, and else format, the user uses boolean literals or expressions that evaluate to a boolean as well as comparison operators to evaluate if statements.

```
var x = "yes"
if(x == "yes"){
print("accepted")
}
else if(x == "no"){
print("denied")
} else {
print("who are you")
}
```

The above returns the string "accepted"

Function Definitions

Functions are defined by including the keyword "function" before your identifier and parameters. The return type is void, unless otherwise stated.

Explicitly Typed

This function uses explicit types for the parameters and returns an int

```
function foo(num: int, truth: bool) : int {
//function operation
return num
} ####Type Inferred This function has a default return type of void. It also uses type inference for
the arguments. function foo(num, truth){
// function operation
}
```

Unary Expressions

Catscript uses the negate (-) unary operator on integers and booleans take the "not" keyword to negate its value

```
var x = 20
print(-x) // This will print "-20"
return not true // This returns false
```

Comparison

Catscript uses the basic comparison operators less than, less than or equal, greater, greater than or equal, as shown below:

```
1 < 0 // false
1 <= 0 // false
```

```
1 > 0 // true
1 >= 0 // true
```

Equality

Catscript also check for equality using the basic equality operators on objects of the same type. The operands used are "equal equal" and "exclamation point equal"

```
1 == 0 // false
1 != 0 // true
```

Variable Statements

Variables are declared using the "var" keyword ####Explicitly Typed You can explicitly define an object's type by including a ": " after var but before your "="

```
var num: int = 8
var hello: string = "Hello World!"
var truth: bool = false
var nums: list<int> = [1,2,3,4,5] ####Inferred Type If you don't explicitly give your variable a type,
the type will be inferred while the statement is parsed var num = 8 //int
var hello = "Hello World!" //string
var truth = false //bool
var nums = [1,2,3,4,5] //list of type int
```

Catscript also allows for lists of type object with multiple dimensions:

```
var list = [8,"Hello World!", [8,40,320]]
```

Print Statements

Catscript print statements are similar to those in other languages where the keyword "print" is used to return data to the user.

```
var num = 8
print(num) //prints out "8"
print("Hello World!") //prints out "Hello World!"
```

You can also use the "+" operator to concatenate integers to your strings

```
var num = 1
print("Hello player " + num)
//returns "Hello player 1" ###Math Operators Catscript uses basic division and multiplication
operators to calculate integers var x = 10
var y = 40
print(x*y) // returns 400
print(x/2) // returns 200
```

Catscript also uses basic addition and subtraction operators

> print(8+40) // evaluates to 48
> print(8-40) // evaluates to -32