Cheatography

Roblox: General Scripting Cheat Sheet

by Ozzypig (Ozzypig) via cheatography.com/25526/cs/6711/

Essential Objects

Class	Description
Part	A physical brick in the world.
Model	A container for Parts.
Folder	A container for Scripts and value objects.
Script	A container for *Lua* source code.
LocalScript	A Script that runs its code on a client.

Basic math functions

Operation	Description
`a + b`	Adds `a` and `b`.
`a - b`	Subtract `a` and `b`.
`a * b`	Multiply `a` and `b`.
`a / b`	Divides `a` by `b`.
`a % b`	Remainder of `a` divided by `b`.
Function	**Description**
`math.random(n)`	Returns random number from `1` to `n` (no negatives).
`math.random(a, b)`	Returns random number from `a` to `b`.
`math.max(...)`	Returns the largest number.
`math.min(...)`	Returns the smallest number.

Basic math functions (cont)

`math.floor(n)`	Rounds `n` down.
`math.ceil(n)`	Rounds `n` up.
`math.abs(n)`	Returns absolute value of `n`.
`math.sqrt(n)`	Returns square root of `n`.
`math.pi`	Approx equal to `3.14159`

It's important to work out problems by hand before translating their solutions into code. Algebra is necessary for success. Read about all math functions here.

String functions

Operation	Description
`a .. b`	Combine two strings.
Function	**Description**
`string.len(str)`	Returns length of `str`.
`string.upper(str)`	Returns `str` in upper-case.
`string.lower(str)`	Returns `str` in lower-case.
`string.reverse(str)`	Returns `str` in reverse.
`string.rep(str, n)`	Returns `str` repeated `n` times

String functions (cont)

`string.sub(str, a, b)`	Return sub-string of `str` from `a` to `b`.

A **string** is a collection of characters, or text. An example of a string property is the `Name` property. Read all string manipulation functions here.

Tables

```
local list = {1, 2, 3}
local firstNum = list[1]
list[2] = 4
print( " There are " .. #list ..
" number s")
local total = 0
for i = 1, #list do
      total = total + list[i]
end
print( "The total is " .. total)
```

Tables are a collection of values. They are defined using curly braces {} with values separated by commas. Access the values inside using square brackets []. Tables are sometimes called **arrays**. Use a **for** loop to work with all items in a table individually. The :GetChildren() method returns a table of children in an object.

By **Ozzypig** (Ozzypig)
cheatography.com/ozzypig/
ozzypig.com

Published 25th January, 2016.
Last updated 24th January, 2017.

Cheatography

Roblox: General Scripting Cheat Sheet

by Ozzypig (Ozzypig) via cheatography.com/25526/cs/6711/

Constants

`game`	Parent of all game services.
`workspace`	Container for all bricks and models are stored.
`script`	The currently running script.

Finding Objects

```
workspace.Part:Destroy()
print( scr ipt.Pa ren t.Name)
game.S erv erS tor age.Tr ee: -
Clone()
```

Use a period to access an object's children. Use .Parent to access an object's parent. Use constants like game, workspace, and script to identify objects in the hierarchy.

Creating objects

How do I create an object?

> Using `Instan ce.n ew (class)` and setting the parent:
> `object.Parent = parent`

How do I access an object's properties?

> Use a period (.):
> `print( obj ect.Name)`

How do I set an object's properties?

> Use a period (.) and equals sign (=):
> `part.T ran spa rency = .5`

How do I destroy an object?

> Using `object :De stroy()`

Creating objects (cont)

How do I copy a preexisting object?

> Using `object :Cl one()` and setting the parent:
> `newTree = workspace.Tree:Clone()`
> `newTree.Parent = workspace`

General Object Functions

Method name	Description
`:FindFirstChild(name)`	Return a child with `na me` or **`nil`** if it doesn't exist.
`:WaitForChild(name)`	Pauses until a child with a name exists and returns it.
`:IsA(className)`	Return whether the object is a certain type of object.
`:Clone()`	Makes and returns a copy of an object.
`:Destroy()`	Permanently delete an object.
`:GetChildren()`	Return a list of an object's children.

These are functions (aka methods) for all classes of ROBLOX objects. Read about all methods here.

Event basics

```
function onTouch(part)
      pri nt( par t.Name .. "
touched me!")
end
worksp ace.Pa rt.T ou che d:c -
onn ect (on Touch)
```

Events are specific occurrences relating to objects. When an event **fires**, or occurs, all connected functions are called.

Basic functions

`wait(n)`	Wait `n` seconds then continue.
`print(...)`	Display something in the Output window.

Variables

```
local myScore = 5
myScore = myScore + 1
print( myS core)
local myName = " Ozz y"
print( "My name is " .. myName)
```

Variables store data of any kind - numbers, strings, tables, objects or **nil** (nothing). A **local** variable is only accessible in the block of code it is defined in.

By **Ozzypig** (Ozzypig)
cheatography.com/ozzypig/
ozzypig.com

Published 25th January, 2016.
Last updated 24th January, 2017.

Cheatography

Roblox: General Scripting Cheat Sheet

by Ozzypig (Ozzypig) via cheatography.com/25526/cs/6711/

If statements

```
if workspace:FindFirstChild("Tree") then
      pri nt( " There is a tree here.")
end
if coins < 5  then
      pri nt( "You need more money." )
else
      pri nt( "You have enough money! ")
end
if player.Name == " Jak e" then
      pri nt( "You are an awesome guy, Jake")
elseif player.Name == " Sal ly" then
      pri nt( "You are a sweeth eart, Sally")
else
      pri nt( "You are a pretty cool person ")
end
```

If statements will run their code if the value between **if**/**then** is true (or not **nil**). They can one an **else** block, or any number of **elseif** blocks.

Loops

Numeric **for** loop

For counting numerically.

Example: Count from `1` to `5`:

```
for i = 1, 5 do
   print(i)
end
```

Generic **for** loop

Most often used for object children.

Example: Print all children in object:

```
for i, child in pairs(object:GetChildren()) do
   print(child.Name)
end
```

While loop

Perform code until a condition is false.

Example: Remove all children named 'Ball'

```
while object:FindFirstChild("Ball") do
   object.Ball:Destroy()
end
```

Repeat-until loop

Perform code once, then again until a condition is true.

Ex.: Copy objects until there are 5.

```
repeat
   newObject = object:Clone()
   newObject.Parent = workspace
   wait(1)
until #workspace:GetChildren() >= 5
```

Loops are used to **iterate**, or repeat code a number of times.

Function examples

```
function sayHello()
      pri nt( " Hello, world")
end
sayHello()
function addTwo Num bers(a, b)
      pri nt( "The sum is:", a + b)
end
addTwo Num bers(3, 5)
function calcul ate Squ are(n)
   return n * n
end
local result = calcul ate Squ - are(3)
```

A function is a named block of code that can be run anywhere in code by **calling** it by name. Functions can have **arguments** (given values) and/or **return** values.

By **Ozzypig** (Ozzypig)
cheatography.com/ozzypig/
ozzypig.com

Published 25th January, 2016.
Last updated 24th January, 2017.