# Lua Small scripting language

## CONCEPTS

Tables are the only data structure: arrays, maps, objects and modules are all tables. Arrays start at index 1.

**Table** your data
↓ `setmetatable`
**Metatable** __index, __add...
↓ `missing key / operator`
**Metamethod** function or table
↓ `result`
**Value** returned

Variables are global unless declared local; typos create new globals silently. Only nil and false are falsy: 0 and "" are true.

## RUN THE INTERPRETER

**`lua <file.lua> [<args>]`**
Run a script with arguments

**`lua`**
Start the interactive REPL

## TYPES AND VARIABLES

**`local <name> = <value>`**
Declare a local variable

**`type(<v>)`**
Name of the value's type

## CONTROL FLOW

**`if <c> then ... elseif <c> then ... else ... end`**
Conditional

**`for i, v in ipairs(<t>) do ... end`**
Array loop, stops at first nil

**`for k, v in pairs(<t>) do ... end`**
All keys, unspecified order

## FUNCTIONS

**`function <name>(<args>) ... end`**
Define a function

## TABLES AND METATABLES

**`local t = {1, 2, 3}`**
Array constructor

**`setmetatable(<t>, <mt>)`**
Attach a metatable, returns t

**`obj:method(<args>)`**
Sugar for obj.method(obj, args)

## STRINGS

**`<s>:find(<pat> [, <init> [, true]])`**
Start and end index; true = plain text

**`<s>:gsub(<pat>, <repl> [, <n>])`**
Replace; returns string and count

## PATTERNS

**`.  %a  %d  %l  %u  %s  %w  %p  %x`**
Any, letter, digit, lower, upper, space, alnum, punct, hex

## TABLE LIBRARY

**`table.insert(<t>, <v>)`**
Append

**`table.remove(<t> [, <pos>])`**
Remove last or at pos, returns it

**`table.concat(<t> [, <sep>])`**
Join strings

## ERRORS

**`assert(<v> [, <msg>])`**
Raise if v is falsy, else return args

**`pcall(<f>, ...)`**
Protected call: true, results or false, err

## MODULES

**`local m = require("<name>")`**
Load a module once, cached

## OS AND I/O

**`io.open(<path>, "r")`**
Open a file; nil, err on failure