Lua: language for the Web?

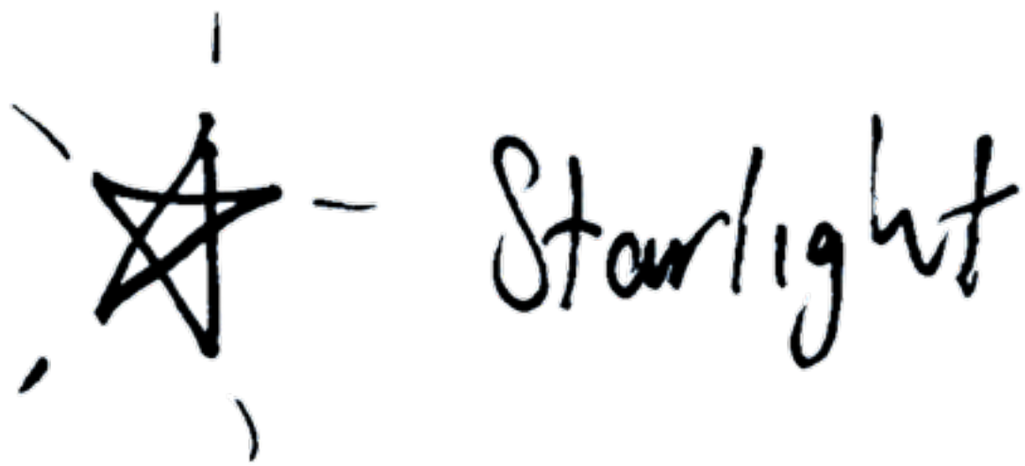
@paulcuth

MOONSHINE

A lightweight Lua VM for the browser

TRY IT GET STARTED

Source available on GitHub.


Starlight

ES6

ES2015

Generators

Syntax

```
function* gen() {
  yield 1;
  yield 2;
  yield 3;
}

var g = gen(); // "Generator { }"
```

MDN MOZILLA DEVELOPER NETWORK

Coroutines

```
co = coroutine.create(function ()
    coroutine.yield(1)
    coroutine.yield(2)
    coroutine.yield(3)
end)

print(coroutine.resume(co))
print(coroutine.resume(co))
print(coroutine.resume(co))
```

Spread operator

Syntax

For function calls:

```
myFunction(...iterableObj);
```

For array literals:

```
[1, 2, 3, ...iterableObj]
```

MDN MOZILLA DEVELOPER NETWORK

table.unpack()

```lua
-- For function calls:
print(table.unpack(t))

-- For table literals:
{ 1, 2, 3, table.unpack(t) }
```

Lua 5.0 Reference Manual

by Roberto Ierusalimschy, Luiz Henrique de Figueiredo, Waldemar Celes

contents · index · other versions

1 – Introduction

Lua is an extension programming language designed to support general procedural programming with data description faciliti
also offers good support for object-oriented programming, functional programming, and data-driven programming. Lua is inte
to be used as a powerful, light-weight configuration language for any program that needs one. Lua is implemented as a lib
written in *clean* C (that is, in the common subset of ANSI C and C++).

Being an extension language, Lua has no notion of a "main" program: it only works *embedded* in a host client, called
embedding program or simply the *host*. This host program can invoke functions to execute a piece of Lua code, can write
read Lua variables, and can register C functions to be called by Lua code. Through the use of C functions, Lua can be augme
to cope with a wide range of different domains, thus creating customized programming languages sharing a syntac
framework.

The Lua distribution includes a stand-alone embedding program, `lua`, that uses the Lua library to offer a complete Lua interpr

Lua is free software, and is provided as usual with no guarantees, as stated in its copyright notice. The implementation desc
in this manual is available at Lua's official web site, `www.lua.org`.

Lua 5.0 Reference Manual

by Roberto Ierusalimschy, Luiz Henrique de Figueiredo, Waldemar Celes

contents · index · other versions

1 – Introduction

Lua is an extension programming language designed to support general procedural programming with data description faciliti
also offers good support for object-oriented programming, functional programming, and data-driven programming. Lua is inte
to be used as a powerful, light-weight configuration language for any program that needs one. Lua is implemented as a lib
written in *clean* C (that is, in the common subset of ANSI C and C++).

Being an extension language, Lua has no notion of a "main" program: it only works *embedded* in a host client, called
embedding program or simply the *host*. This host program can invoke functions to execute a piece of Lua code, can write
read Lua variables, and can register C functions to be called by Lua code. Through the use of C functions, Lua can be augme
to cope with a wide range of different domains, thus creating customized programming languages sharing a syntac
framework.

The Lua distribution includes a stand-alone embedding program, `lua`, that uses the Lua library to offer a complete Lua interpr

Lua is free software, and is provided as usual with no guarantees, as stated in its copyright notice. The implementation desc
in this manual is available at Lua's official web site, `www.lua.org`.

2003

http://www.katalog-telefonow-komorkowych.pl/files/mobily/st1/1111/800x600_pl_PL/1330689341-orange-spv-e200-1.jpg

2003

http://www.okokchina.com/Files/uppic6/Orange%20SPV%20GSM%20Mobile%20phone%20w_clip%20on%20camera847.jpg

Rest operator

Syntax

```
function(a, b, ...theArgs) {
  // ...
}
```

MDN MOZILLA DEVELOPER NETWORK

Varargs

```
function(a, b, ...)
    -- ...
end
```

Reference Manual of the Programming Language Lua 2.5

Roberto Ierusalimschy, Luiz Henrique de Figueiredo, Waldemar Celes
lua@tecgraf.puc-rio.br
TeCGraf, Computer Science Department, PUC-Rio

Abstract. Lua is an extension programming language designed to be used as a configuration language for any program that needs one. This document describes version 2.5 of the Lua programming language and the API that allows interaction between Lua programs and their host C programs.

Sumário. Lua é uma linguagem de extensão projetada para ser usada como linguagem de configuração em qualquer programa que precise de uma. Este documento descreve a versão 2.5 da linguagem de programação Lua e a Interface de Programação (API) que permite a interação entre programas Lua e programas C hospedeiros.

Last update: Thu Jun 26 08:15:42 EST 1997 by lhf.

1996

http://www.billboard.com/files/styles/article_main_image/public/media/spice-girls-1996-billboard-650.jpg

Proxies

```
var handler = {
    get: function(target, name){
        return name in target?
            target[name] :
            37;
    }
};

var p = new Proxy({}, handler);
p.a = 1;
p.b = undefined;

console.log(p.a, p.b); // 1, undefined
console.log('c' in p, p.c); // false, 37
```

MDN MOZILLA DEVELOPER NETWORK

4.7 Fallbacks

Lua provides a powerful mechanism to extend its semantics, called *fallbacks*. Basically, a fallback is a programmer defined function which is called whenev
proceed.

Lua supports the following fallbacks, identified by the given strings:

``arith"
called when an arithmetic operation is applied to non numerical operands, or when the binary ^ operation is called. Receives three arguments: the two
the operation is unary minus) and one of the following strings describing the offended operator:

```
add  sub  mul  div  pow  unm
```

Its return value is the final result of the arithmetic operation. The default function issues an error.

``order"
called when an order comparison is applied to non numerical or non string operands. Receives three arguments: the two operands and one of the follo
operator:

```
lt gt le ge
```

Its return value is the final result of the comparison operation. The default function issues an error.

``concat"
called when a concatenation is applied to non string operands. Receives the two operands as arguments. Its return value is the final result of the conca
function issues an error.

``index"
called when Lua tries to retrieve the value of an index not present in a table. Receives as arguments the table and the index. Its return value is the fina
default function returns nil.

``gettable"
called when Lua tries to index a non table value. Receives as arguments the non table value and the index. Its return value is the final result of the ind
issues an error.

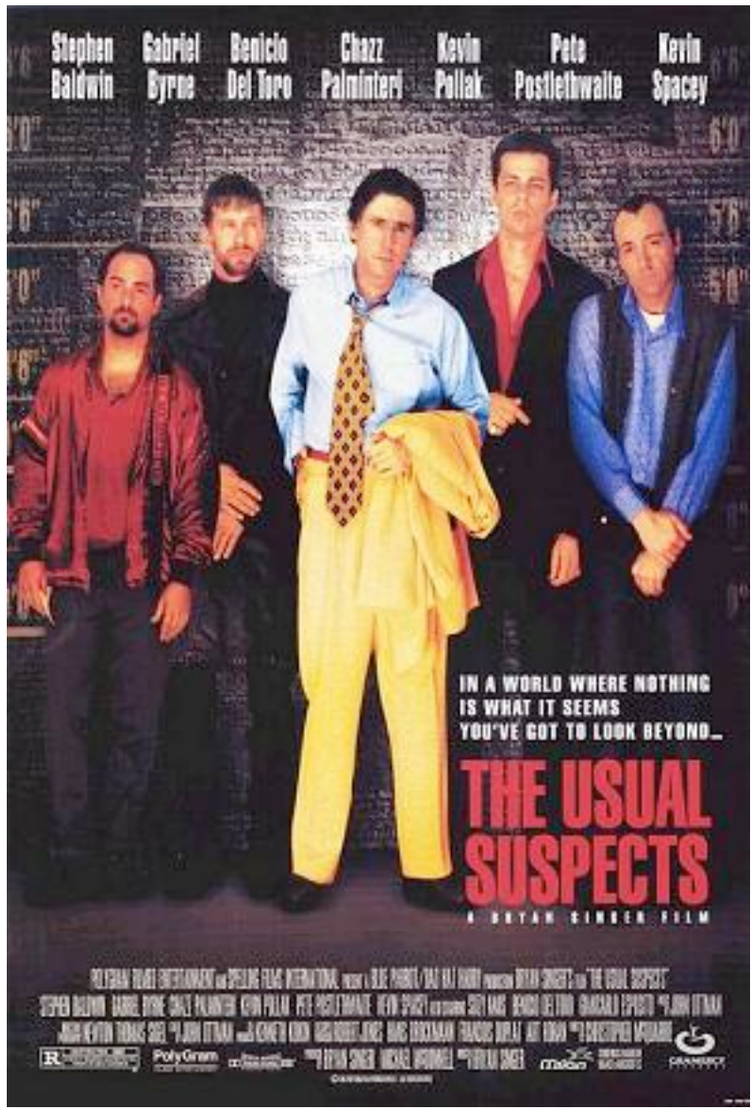
``settable"
called when Lua tries to assign indexed a non table value. Receives as arguments the non table value, the index, and the assigned value. The default f

``function"
called when Lua tries to call a non function value. Receives as arguments the non function value and the arguments given in the original call. Its retur
call operation. The default function issues an error.

``gc"
called during garbage collection. Receives as argument the table being collected. After each run of the collector this function is called with argument
during garbage collection, it must be used with great care, and programmers should avoid the creation of new objects (tables or strings) in this functio

``error"
called when an error occurs. Receives as argument a string describing the error. The default function prints the message on the standard error output

1995



https://en.wikipedia.org/wiki/The_Usual_Suspects#/media/File:Usual_suspects_ver1.jpg

1995

https://upload.wikimedia.org/wikipedia/en/1/13/Toy_Story.jpg

1995

https://en.wikipedia.org/wiki/Sailor_Moon_Super_S:_The_Movie#/media/File:Sailor_Moon_Super_S_The_Movie_poster.jpg

Destructuring assignments

```
1  var a = 1;
2  var b = 3;
3
4  [a, b] = [b, a];
```

MDN MOZILLA DEVELOPER NETWORK

Multiple return values

```
function f() {
  return [1, 2];
}
```

MDN MOZILLA DEVELOPER NETWORK

Block scoping

```
1  if (x > y) {
2    let gamma = 12.7 + y;
3    i = gamma * x;
4  }
```

MDN MOZILLA DEVELOPER NETWORK

Reference Manual of the Programming Language Lua

Roberto Ierusalimschy
Luiz Henrique de Figueiredo
Waldemar Celes Filho

TeCGraf — PUC-Rio
roberto, lhf, celes@icad.puc-rio.br

May 27, 1994

Abstract

Lua is an embedded programming language designed to be used as a configuration language for any program that needs one. This document describes the Lua programming language and the API that allows interaction between Lua programs and its host C program. It also presents some examples of using the main features of the system.

Sumário

Lua é uma linguagem de extensão projetada para ser usada como linguagem de configuração em qualquer programa que precise de uma. Este documento descreve a linguagem de programação Lua e a Interface de Programação que permite a interação entre programas Lua e o programa C hospedeiro. O documento também apresenta alguns exemplos de uso das principais características do sistema.

1994

https://upload.wikimedia.org/wikipedia/ca/6/68/FIFA_World_Cup_1994_Logo.png

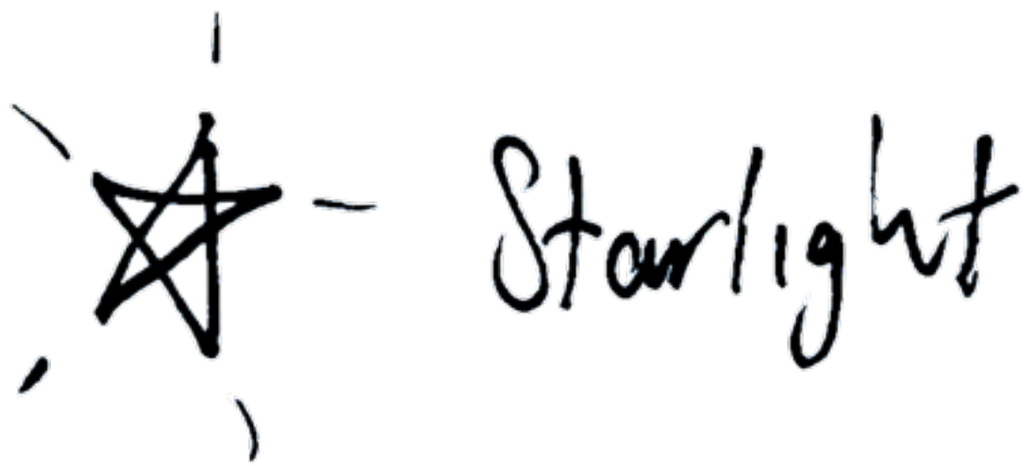
1994

http://soccernet-assets.espn.go.com/design05/i/world%20cup%20images/win_g_1994_576.jpg

21 years

http://www.famousbirthdays.com/year/1994.html

ES6


Starlight

Hello Web

```
<!DOCTYPE html>
<html>
  <body>
    <h1>Starlight</h1>

    <script type="application/x-lua">
      print 'Hello Web'
    </script>

    <script src="//cdnjs.cloudflare.com/ajax/libs/babel-core/5.8.34/browser.min.js"></script>
    <script src="../lib/starlight.js" data-run-script-tags></script>

  </body>
</html>
```

Configuration

```
<p id="output"></p>

<script>
  window.starlight = {
    config: {
      stdout: {
        writeln: function (message) {
          document.getElementById('output').innerHTML += message + '<br/>';
        }
      },
      env: {
        getTimestamp: Date.now.bind(Date)
      }
    }
  };
</script>

<script type="application/x-lua">
  print 'Hello Web'
  print('now: '..getTimestamp())
</script>
```

Modules

```
<script type="application/x-lua" data-modname="greeting">
  return {
    greet = function(name)
      print('Hello '..name)
    end
  }
</script>

<script type="application/x-lua">
  local greeting = require 'greeting'
  greeting.greet 'FOSDEM'
</script>
```

DOMAPI

```
<script type="application/x-lua">
  local div = window.document:querySelector 'div'

  function handleClick ()
    for frame = 0, 7 do
      window:setTimeout(function ()
        div.className = 'frame-'..frame
      end, 50 * frame)
    end
  end

  window.document:addEventListener('click', handleClick)
</script>
```

window.extract()

```
<script type="application/x-lua">
  window.extract();
  local div = document:querySelector 'div'

  function handleClick ()
    for frame = 0, 7 do
      setTimeout(function ()
        div.className = 'frame-'..frame
      end, 50 * frame)
    end
  end

  document:addEventListener('click', handleClick)
</script>
```

Grunt task

```
npm install grunt-starlight --save-dev
```

Grunt task

```
grunt.initConfig({
  starlight: {
    trails: {
      src: 'src/script/trails.lua',
      dest: 'dist/script/trails.es6.js',
    }
  },
  browserify: {
    options: {
      transform: ['babelify'],
    },
    trails: {
      files: {
        'dist/script/trails.js': ['dist/script/trails.es6.js'],
      },
    },
  },
});
```

Roadmap

- Source mapping
- Gulp task
- <script type="application/x-lua" src="…">
- Plug-in system (coroutines)

Please help

https://github.com/paulcuth/starlight

@paulcuth

Questions?

https://github.com/paulcuth/starlight

@paulcuth