Introduction to
Mobile App Development

MODULE 2:
Introduction to Lua

Module 2

1. What is Lua?
2. Conventions
3. Types and Values
4. Tables
5. Expressions
6. Functions

3

What is Lua?

Programming Language

- Lua is an extension programming language designed to support general procedural programming with data description facilities.
- Lua is intended to be used as a powerful, lightweight scripting language for any program that needs one.

4

5

Conventions

Names

- Names (also called identifiers) in Lua can be any string of letters, digits, and underscores, not beginning with a digit.
- This coincides with the definition of names in most languages. The definition of "letter" depends on the current locale: any character considered alphabetic by the current locale can be used in an identifier. Identifiers are used to name variables and table fields.
- The following keywords are reserved and cannot be used as names:

and	**break**	**do**	**else**	**elseif**	**end**	**false**
for	**function**	**if**	**in**	**local**	**nil**	**not**
or	**repeat**	**return**	**then**	**true**	**until**	**while**

6

Conventions

Names

- Lua is a case-sensitive language: ***and*** is a reserved word, but **And** and **AND** are two different, valid names.
- As a convention, names starting with an underscore followed by uppercase letters (such as **_VERSION**) are reserved for internal global variables used by Lua.
- A comment starts with a double hyphen (--) anywhere outside a string. They run until the end of the line. You can comment out a full block of code by surrounding it with --[[and --]]. To uncomment the same block, simply add another hyphen to the first enclosure, as in ---[[.

```
-- Single line commented out

--[[ Entire block commented out
print( 10 )
print( 15 )
--]]
```

7

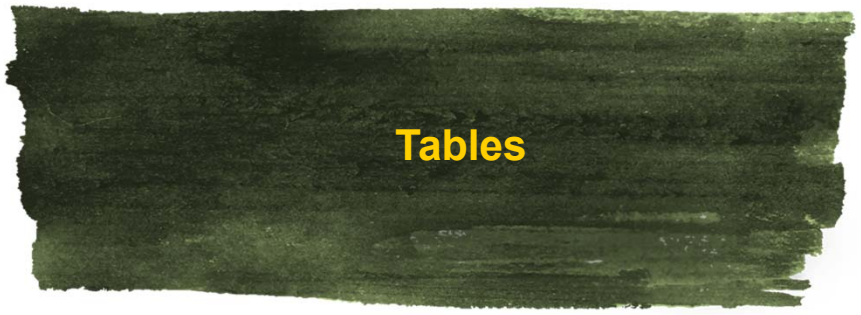
8

Types and Values

Lua scripting

- Lua is a **dynamically typed language**. This means that variables do not have types; only values do.
- The basic types you should be concerned with are:
  - **nil** — the type of the value nil, whose main property is to be different from any other value; it usually represents the absence of a useful value.
  - **boolean** — the type of the values false and true. Both nil and false make a condition false; any other value makes it true.
  - **number** — represents real (double-precision floating-point) numbers.
  - **string** — represents arrays of characters. Lua is 8-bit clean: strings can contain any 8-bit character, including embedded zeros.
  - **function** — represents a section of a program that performs a specific task.
  - **table** — the sole data structuring mechanism in Lua.

9



10

Tables

Lua scripting

- Tables are the only data structure available in Lua that helps us create different types like arrays and dictionaries.

colors

Index/key	value
1	blue
2	red
3	green

11

Tables

Table Manipulation

- There are in built functions for table manipulation and they are listed in the following table.

S.N.	Method & Purpose
1	**table.concat (table [, sep [, i [, j]]])** Concatenates the strings in the tables based on the parameters given.
2	**table.insert (table, [pos,] value)** Inserts a value into the table at specified position.
3	**table.maxn (table)** Returns the largest numeric index.
4	**table.remove (table [, pos])** Removes the value from the table.
5	**table.sort (table [, comp])** Sorts the table based on optional comparator argument.

12

Tables

Table Concatenation

- Example

```
Concatenated string   blueredgreen
Concatenated string   blue, red, green
Concatenated string   red, green
```

```
colors = {"blue","red","green"}

-- returns concatenated string of table
print("Concatenated string ",table.concat(colors))

--concatenate with a character
print("Concatenated string ",table.concat(colors,", "))

--concatenate colors based on index
print("Concatenated string ",table.concat(colors,", ", 2,3))
```

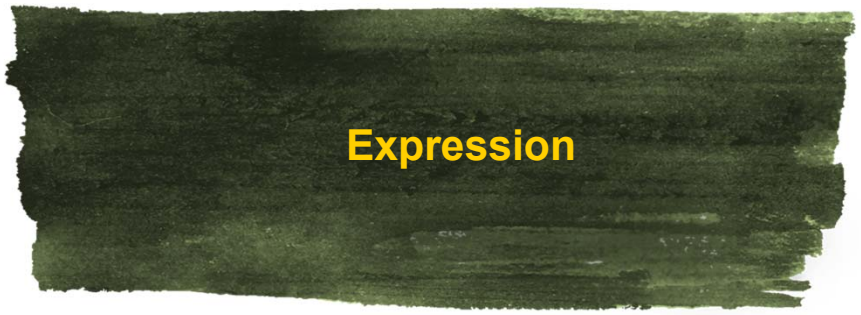

Tables

Insert and Remove

- Example

```
color at index 4 is   yellow
color at index 2 is   pink
The maximum elements in table is   5
The last element is   yellow
The maximum elements in table is   4
```

```
colors = {"blue","red","green"}

-- insert a color at the end
table.insert(colors,"yellow")
print("color at index 4 is ",colors[4])

--insert color at index 2
table.insert(colors,2,"pink")
print("color at index 2 is ",colors[2])

print("The maximum elements in table is",table.maxn(colors))

print("The last element is",colors[5])

table.remove(colors)

print("The maximum elements in table is",table.maxn(colors))
```



15

Expression

Lua scripting

- **Arithmetic expressions:** Lua has the usual arithmetic operators.
- **Relational expressions:**
  Relational operators are supplied which return the boolean values true or false.
  - == equal to
  - ~= not equal to
  - < less than
  - > greater than
  - <= less than or equal to
  - >= greater than or equal to

16

17

Function

Lua scripting

- Functions are the main mechanism for abstraction of statements and expressions in Lua.
- Functions can both carry out a specific task (what is sometimes called procedure or subroutine in other languages) or compute and return values.
- In the first case, we use a function call as a statement; in the second case, we use it as an expression:

Examples:

```
print(8*9, 9/8)

a = math.sin(3) + math.cos(10)

print(os.date())
```

```
13:01:45.501  72    1.125
13:01:45.501  -0.69795152101659
13:01:45.501  04/17/17 13:01:45
```

18

Function

Lua scripting: Factorial Example


```
-- defines a factorial function
   function fact (n)
    if n == 0 then
      return 1
    else
      return n * fact(n-1)
    end
   end

    a = 4
    x = fact(a)
   local mytext = display.newText("factorial(" .. a .. ")" .. " = " .. x, 150, 150, native.systemFont, 20)
   mytext:setTextColor(0, 255, 0)
```

19

End of Module 2