LUA Metatables

Role of metatables

Metatables can be assigned to LUA tables to describe or change behaviour of operations on LUA tables. If table has metatable assigned, LUA engine looks for some special fields in metatables if some special events occurs. For example, if somebody wants to add two tables (system doesn't know how to add two tables because content is user dependent), LUA engine looks for special key **__add** in metatable to execute operator "**+**" and tries to execute funtion assigned to **__add** variable. In this case **__add** is a so called **metamethod**.

Possible LUA metamethods in metatable (see MetatableEvents):

- Arithmetic: **__unm, __add, __sub, __mul, __div, __mod, __pow, __concat**
- Comparison: **__eq, __lt, __le**
- Special: **__index, __newindex, __mode, __call, __metatable, __tostring, __len, ~~__gc~~**

Assignment of metatables

By default LUA table has no metatable assigned.

```
local NormalTable = {}
print(getmetatable(NormalTable))
-- will print "nil"
```

Metatables are normal LUA tables

```
local NormalTable = {}
local MetaTable = {}
setmetatable (NormalTable, MetaTable)
print(getmetatable(NormalTable))
-- will print: table: 0x20db880
```

If mettables are normal table, so it is possible to use the same table also as metatable to save some resources

```
setmetatable (NormalTable, NormalTable)
```

setmetatable(table, metatable) returns its first argument, so it is possible to simplify code below:

```
local NormalTable = {}
setmetatable (NormalTable, NormalTable)
```

to

```
local NormalTable = setmetatable ({}, {})
```

Metamethods

http://lua-users.org/wiki/MetatableEvents

http://lua-users.org/wiki/MetamethodsTutorial

- **__index** - is called when key in table is accessed. **__index** can be a function or another table - fallback table. Fallback table can have metatable which points to another fallback table and so on. It is possible to create very long fallback table chain.
  - **__index = function (table, key)**, return value of fucntion is returned as result.
  - **__index = table**
  - to access table key without invoking __index methamethod use **rawget(table,key)**
- * **__newindex** is called when new value is assigned to key key
  - **__newindex = function (table, key ,value)**
  - to set new value without invoking methamethod, use **rawset(table, key, value)**
- **__metatable** when set, metatable is hidden. Value of **__metatable** is returned instead of original metatable.
- **__call** - if somebady calls table as function, this metamethod will be called
- **__tostring** - when **tostring(table)** is called
- **__len**

Synctatic sugar

To satisfy programmers and increase code readability function can be declared using colon ":" syntax

Usual way of declaring function:

```
function class.method(self, a, b, c)
  -- ...
end
```

can be declared also as

```
function class:method(a, b, c)
  -- ...
end
```

Function calls are also possible using both methods:

```
class:method(1, 2, 3)
class.method(t, 1, 2, 3)
```