Módulo JSON

O módulo **json** fornece funções para converter entre dados Lua e formato JSON (JavaScript Object Notation). É útil para comunicação com APIs web, armazenamento de configurações, serialização de dados e interoperabilidade com outros sistemas.

Funções Disponíveis

1. `json.encode(value)`

Codifica um valor Lua em uma string JSON.

Parâmetros:

- `value` (qualquer tipo): Valor a ser codificado para JSON.

Retorno:

- `string`: Representação JSON do valor

Exceções:

- Lança erro se o valor contiver referências circulares
- Lança erro se a tabela tiver chaves de tipos mistos
- Lança erro se não for possível serializar algum tipo de dado

Exemplos:

```
-- Codificar valores básicos
local json_null = json.encode(nil)           -- "null"
local json_bool = json.encode(true)          -- "true"
local json_num = json.encode(42.5)           -- "42.5"
local json_str = json.encode("Hello\nWorld") -- "\"Hello\\nWorld\""

-- Codificar array (tabela com índices numéricos sequenciais)
local array = {"apple", "banana", "orange"}
local json_array = json.encode(array)
```

```
-- Resultado: "[\"apple\",\"banana\",\"orange\"]"

-- Codificar objeto (tabela com chaves de string)
local person = {
    name = "João Silva",
    age = 30,
    active = true,
    tags = {"developer", "backend"},
    address = {
        street = "Rua das Flores, 123",
        city = "São Paulo"
    }
}
local json_person = json.encode(person)
-- Resultado: {"name":"João
Silva","age":30,"active":true,"tags":["developer","backend"],"address":{"street":"Rua das
Flores, 123","city":"São Paulo"}}

-- Codificar dados de monitoramento
local metrics = {
    timestamp = os.time(),
    hostname = "server-01",
    cpu_usage = 45.7,
    memory_mb = 2048,
    services = {"nginx", "postgresql", "redis"},
    status = "healthy"
}
local json_metrics = json.encode(metrics)

-- Codificar lista de eventos
local events = {
    {
        id = 1,
        type = "login",
        user = "admin",
        timestamp = "2024-01-15T10:30:00Z"
    },
    {
        id = 2,
        type = "logout",
```

```
        user = "user1",
        timestamp = "2024-01-15T11:45:00Z"
    }
}
local json_events = json.encode(events)
```

2. `json.decode(string)`

Decodifica uma string JSON em um valor Lua de forma assíncrona.

Parâmetros:

- `string` (string): String JSON a ser decodificada

Retorno:

- `any`: Objeto decodificado como um valor Lua (mais frequentemente uma tabela):

Exceções:

- Lança erro se a string não for JSON válido
- Lança erro se houver profundidade excessiva de aninhamento
- Lança erro se números forem muito grandes ou muito pequenos

Exemplos:

```
-- Decodificar valores básicos
local null_val = json.decode("null")           -- nil
local bool_val = json.decode("true")           -- true
local num_val = json.decode("42.5")            -- 42.5
local str_val = json.decode("\"Hello\"")       -- "Hello"

-- Decodificar array JSON
local json_array = "[\"apple\", \"banana\", \"orange\"]"
local array = json.decode(json_array)
-- Resultado: {"apple", "banana", "orange"}
print(array[1])  -- "apple"
print(array[2])  -- "banana"
print(#array)    -- 3

-- Decodificar objeto JSON
local json_person = [[
{
```

```
    "name": "Maria Santos",
    "age": 28,
    "active": true,
    "skills": ["Python", "Lua", "JavaScript"],
    "metadata": {
        "department": "Engineering",
        "level": "Senior"
    }
}
]]
local person = json.decode(json_person)
-- Resultado: tabela com chaves name, age, active, skills, metadata
print(person.name)                    -- "Maria Santos"
print(person.age)                     -- 28
print(person.skills[1])               -- "Python"
print(person.metadata.department)     -- "Engineering"

-- Decodificar resposta de API
local api_response = [[
{
    "status": "success",
    "data": {
        "users": [
            {"id": 1, "name": "Alice", "email": "alice@example.com"},
            {"id": 2, "name": "Bob", "email": "bob@example.com"}
        ],
        "total": 2,
        "page": 1
    },
    "timestamp": "2024-01-15T12:00:00Z"
}
]]
local response = json.decode(api_response)
if response.status == "success" then
    for _, user in ipairs(response.data.users) do
        print("Usuário: " .. user.name .. " (" .. user.email .. ")")
    end
    print("Total: " .. response.data.total)
end

-- Decodificar configurações
```

```
local config_json = [[
{
    "server": {
        "host": "0.0.0.0",
        "port": 8080,
        "timeout": 30
    },
    "database": {
        "host": "localhost",
        "name": "monagent",
        "pool_size": 10
    },
    "logging": {
        "level": "info",
        "file": "/var/log/monagent.log"
    }
}
]]
local config = json.decode(config_json)
local server_host = config.server.host          -- "0.0.0.0"
local db_pool = config.database.pool_size       -- 10
local log_level = config.logging.level          -- "info"
```

Revision #1
Created 3 February 2026 16:43:17 by Marc
Updated 3 February 2026 16:43:33 by Marc