Módulo WS

O módulo **ws** fornece funcionalidades para comunicação bidirecional em tempo real através do protocolo WebSocket. Este módulo é útil para scripts que precisam se conectar a serviços que utilizam WebSockets.

Características principais:

- Conexão WebSocket assíncrona
- Envio e recebimento de mensagens de texto
- Suporte a URLs seguras (wss://) e não seguras (ws://)
- Timeout implícito baseado no sistema

Protocolo WebSocket:

- Protocolo de comunicação full-duplex sobre uma única conexão TCP
- Ideal para aplicações em tempo real
- Suportado pela maioria dos navegadores e servidores modernos
- Menor overhead comparado a HTTP polling

Funções Disponíveis

1. `ws.send_recv(url, dados)`

Estabelece uma conexão WebSocket, envia uma mensagem e aguarda uma resposta.

Parâmetros:

- **url** (string): URL do servidor WebSocket (ex: "ws://exemplo.com/socket", "wss://exemplo.com/ws")
- **dados** (string): Mensagem de texto a ser enviada para o servidor

Retorno:

- **string**: Resposta de texto recebida do servidor WebSocket

Comportamento:

1. Estabelece conexão WebSocket com o servidor especificado

2. Envia a mensagem de texto fornecida
3. Aguarda uma resposta do servidor
4. Retorna a primeira mensagem de texto recebida
5. Fecha a conexão após receber a resposta
6. Lança erro se receber mensagem binária ou se não houver resposta

Erros Comuns:

- `"ws binary messages not supported"` - Servidor enviou mensagem binária (não suportada)
- `"no response from web socket"` - Servidor não respondeu
- Erros de conexão (servidor offline, URL inválida, etc.)

Exemplo de Uso:

```
-- Conexão básica com servidor WebSocket
local url = "ws://echo.websocket.org"
local mensagem = "Olá, WebSocket!"
local resposta = ws.send_recv(url, mensagem)
print("Resposta do servidor:", resposta)
-- Saída: "Olá, WebSocket!" (servidor echo)

-- Conexão segura (wss://)
local url_segura = "wss://servidor.producao.com/ws"
local dados = json.encode({acao = "ping", timestamp = now()})
local resposta = ws.send_recv(url_segura, dados)
print("Resposta segura:", resposta)

-- Com tratamento de erro
local function enviar_com_tratamento(url, dados)
    local ok, resposta = pcall(ws.send_recv, url, dados)
    if ok then
        return resposta
    else
        log.error("Erro no WebSocket:", resposta)
        return nil
    end
end

-- Testar múltiplos servidores
local servidores = {
    "ws://servidor1.com/ws",
```

```
    "ws://servidor2.com/socket",
    "ws://backup.servidor.com/ws"
}

for _, servidor in ipairs(servidores) do
    local resposta = enviar_com_tratamento(servidor, "ping")
    if resposta then
        print("Servidor", servidor, "respondendo")
        break
    end
end
```

Informações Adicionais

Conexão Segura (wss://):

```
-- O módulo suporta tanto ws:// quanto wss://
local conexoes = {
    "ws://localhost:8080/ws",      -- Não seguro (HTTP)
    "wss://servidor.com/ws",       -- Seguro (HTTPS)
    "ws://192.168.1.100:3000/ws",  -- Local network
}

for _, url in ipairs(conexoes) do
    local ok, resposta = pcall(ws.send_recv, url, "ping")
    if ok then
        print("Conexão bem-sucedida:", url)
    else
        print("Falha na conexão:", url, "-", resposta)
    end
end
```

Mensagens de Texto Apenas:

```
-- O módulo só suporta mensagens de texto
-- Mensagens binárias retornam erro
```

```
local function enviar_dados_seguros(url, dados)
    -- Converter dados para JSON (texto)
    local dados_json = json.encode(dados)

    local resposta = ws.send_recv(url, dados_json)

    -- Parsear resposta JSON
    return json.decode(resposta)
end

-- Exemplo com dados complexos
local dados_complexos = {
    usuarios = {
        {id = 1, nome = "Alice", ativo = true},
        {id = 2, nome = "Bob", ativo = false}
    },
    metricas = {
        cpu = 45.6,
        memoria = 78.3,
        timestamp = now()
    }
}

local resposta = enviar_dados_seguros("wss://api.empresa.com/ws", dados_complexos)
```

Revision #1
Created 3 February 2026 16:48:20 by Marc
Updated 3 February 2026 16:48:34 by Marc