PORADNIK LUA

Cygnus Reborn MUD Online Client Engine

> Ten dokument zawiera kompletne podstawy języka Lua oraz dedykowanego API klienta gry Cygnus Reborn.
> Skrypty możesz uruchamiać w **Triggerach**, **Aliasach** lub bezpośrednio z konsoli klienta wpisując: `lua: <kod>` lub `#lua <kod>` .

1. WYPISYWANIE WIADOMOŚCI (LOKALNE ECHO)

Funkcje wypisują komunikat bezpośrednio na ekran terminala (widoczny tylko dla Ciebie, nie jest wysyłany do serwera gry).

```lua
-- echo(tekst) lub print(tekst) generuje lokalne powiadomienie
echo("Witaj w silniku Lua!")
print("To też zadziała i wypisze komunikat na zielono.")
```

2. WYSYŁANIE KOMEND DO GRY

Podstawowe narzędzie do interakcji ze światem MUD za pomocą skryptów.

```lua
-- send(komenda) wysyła wpisany tekst bezpośrednio do serwera MUD
send("look")
send("ask Cześć wszystkim!")
```

3. ZMIENNE GLOBALNE I ICH SYNCHRONIZACJA

Klient posiada globalną tabelę g, która automatycznie zapisuje stan w pamięci podręcznej przeglądarki.

```lua
-- Odczyt zmiennej z globalnego repozytorium 'g':
local obecny_cel = g.cel
echo("Mój obecny cel to: " .. tostring(obecny_cel))

-- Zapis/Modyfikacja zmiennej (zostanie automatycznie zapamiętana):
g.cel = "ork"
g.bron = "miecz"
echo("Zmieniono cel na: " .. g.cel)

-- Możesz też użyć alternatywnej funkcji pomocniczej set():
set("cel", "smok")
```

4. DOPASOWANIA TRIGGERÓW (ZMIENNA 'MATCHES')

Po wywołaniu triggera, w skrypcie automatycznie dostępna jest tabela matches (analogicznie do systemu Mudlet).

```
-- matches[1] - Cała dopasowana linia tekstu
-- matches[2] - Pierwsza złapana grupa z wyrażenia regularnego (np. z nawiasu (.*))
-- matches[3] - Druga złapana grupa regex, itd.

-- Przykład dla wzoru triggera: ^(Moneta|Złoto) wypada z (.*)\.$
local co_wypadlo = matches[2]  -- Wynik: "Moneta" lub "Złoto"
local z_kogo = matches[3]      -- Wynik: np. "orka"
send("wez " .. co_wypadlo .. " z " .. z_kogo)
```

5. OPERACJE NA TEKŚCIE (STRINGI)

Wiązanie ciągów tekstowych oraz podstawowe manipulacje tekstowe w standardzie Lua.

```
-- Łączenie tekstów odbywa się za pomocą operatora '..'
local imie = "Kael"
local powitanie = "Witaj, " .. imie .. "!"
echo(powitanie)

-- Przydatne funkcje biblioteki string:
local dlugosc = string.len(imie)         -- Wynik: 4
local duze_litery = string.upper(imie)   -- Wynik: "KAEL"
local male_litery = string.lower(imie)   -- Wynik: "kael"

-- Wycinanie fragmentów (podciągów):
local sub = string.sub("Cygnus", 1, 3)    -- Wynik: "Cyg"
```

6. TABLICE I SŁOWNIKI (TABLES)

Główna struktura danych w Lua. Pamiętaj: tablice tradycyjnie indeksujemy od 1!

```lua
-- A. Prosta tablica (indeksowana liczbowo):
local kierunki = {"polnoc", "poludnie", "wschod", "zachod"}
echo("Pierwszy kierunek: " .. kierunki[1]) -- "polnoc"

-- Dodawanie elementu na koniec tablicy:
table.insert(kierunki, "gora")

-- B. Słownik (struktura klucz-wartość):
local postac = {
    imie = "Garr",
    klasa = "Wojownik",
    poziom = 15
}
echo(postac.imie .. " jest " .. postac.klasa)

-- Pętla po słowniku (klucze nieuporządkowane):
for klucz, wartosc in pairs(postac) do
    echo(klucz .. " = " .. tostring(wartosc))
end

-- Pętla po tablicy (indeksowana i uporządkowana):
for i, kierunek in ipairs(kierunki) do
    echo("Kierunek " .. i .. ": " .. kierunek)
end
```

7. INSTRUKCJE WARUNKOWE I LOGIKA

Kontrola przepływu skryptów automatyzacji walki lub obrony.

```lua
local hp = 45
local max_hp = 100

if hp < 30 then
    send("czar leczenie")
    echo("Ostrzeżenie: Niskie punkty życia!")
elseif hp < 70 then
    echo("Zdrowie w normie.")
else
    echo("Pełnia sił!")
end

-- Operatory logiczne: and, or, not
if hp < 50 and not (g.cel == "smok") then
    send("say Dam sobie radę!")
end
```