NAME

akfavatar.utf8 − Module für UTF-8 Unterstützung in Lua-AKFAvatar

SYNTAX

local utf8 = require "akfavatar.utf8"

BESCHREIBUNG

Dieses Modul definiert Funktionen für UTF-8 Strings. Viele dieser Funktionen dienen als Ersatz für Funktionen in Luas String-Bibliothek.

UTF-8 ist eine Zeichen-Kodierung für Unicode. Ein Zeichen kann mit einem bis vier Byte kodiert sein. Darum wird eine besondere Behandlung erforderlich. Die meisten Funktionen der String-Bibliothek von Lua können nämlich nur mit Kodierungen mit einem Byte pro Zeichen umgehen.

utf8.len(*String***)**
: Zählt die Anzahl der Zeichen in einem UTF-8 kodierten *String*.
Hinweis: Steuerzeichen und unsichtbare Zeichen werden auch gezählt.

utf8.sub(*String, Anfangszeichen [,Endzeichen]***)**
: Wie **string.sub**, aber für UTF-8 Strings.

Gibt den Teilstring von *Anfangszeichen* bis *Endzeichen* zurück. Falls *Anfangszeichen* oder *Endzeichen* negativ sind, dann werden sie vom Ende des Strings gezählt.
Also, utf8.sub(s, 1, 3) gibt die ersten 3 Zeichen zurück, während utf8.sub(s, -3) die letzten 3 Zeichen zurück gibt.

utf8.char(*...***)**
: Wie **string.char**, aber es akzeptiert höhere Werte und gibt einen UTF-8 kodierten String zurück.

utf8.codepoint(*String***)**
: Gibt den Codepoint des ersten Zeichens des *String*s zurück.
Im Falle eines Fehlers gibt es *nil* zurück (aber das ist keine richtige Gültigkeitsprüfung).

utf8.codepoints(*String [,Anfangszeichen [,Endzeichen]]***)**
: Wie **string.byte**.
Gibt die Unicode-Werte von *Anfangszeichen* bis *Endzeichen* zurück.
Wenn man nur das erste Zeichen benötigt, sollte man stattdessen **utf8.codepoint()** verwenden.

utf8.characters(*String***)**
: Iterator für die einzelnen Zeichen eines UTF-8 Strings.
Ein Zeichen kann ein Einzelbyte oder Multibyte-String sein.

Anwendungsbeispiel:

```
for c in utf8.characters(line) do print(utf8.codepoint(c)) end
```

utf8.reverse(*String***)**
: Dreht einen UTF-8 *String* um.
Hinweis: Kombinationszeichen sind noch problematisch.

utf8.rep(*String, n***)**
: Gibt den *String n* mal wiederholt zurück. Dies ist nur ein Alias für **string.rep()**.

utf8.underlined(*String***)**
: Gibt den *String* unterstrichen zurück (Overstrike-Technik).

utf8.bold(*String***)**
: Gibt den *String* fett gedruckt zurück (Overstrike-Technik).

utf8.bom
: Byte Order Mark.
Das wird bei UTF-8 zwar nicht benötigt, wird aber manchmal als Signatur eingesetzt.

utf8.check_bom(*String***)**
: Überprüft, ob der *String* mit einem UTF-8-BOM anfängt.

utf8.check(*String***)**
: Überprüft, ob der *String* in UTF-8 kodiert ist.
Es ist nur eine Überprüfung ob UTF-8 oder nicht, es ist keine Gültigkeitsüberprüfung.
Hinweis: reines ASCII ist auch gültiges UTF-8.

utf8.check_unicode(*String***)**
: Überprüft den String auf Unicode-Kodierungen.
Gibt eines von "UTF-8", "UTF-16BE", UTF-16LE", "UTF-32BE", "UTF-32LE" zurück, oder *nil*, wenn keine Unicode Kodierung erkannt wird.

utf8.from_ncr(*String***)**
: Ersetzt Numeric Character References (NCR) mit UTF-8 Zeichen.
Zum Beispiel "€" (dezimal) oder "€" (hexadezimal) für das Euro Währungszeichen.

utf8.to_ncr(*String***)**
: Ersetzt nicht-ASCII Zeichen mit NCRs. Das Ergebnis ist ein reiner ASCII-String, aber kodiert.

utf8.from_latin1(*String***)**
: Konvertiert einen *String* von Latin-1 (ISO-8859-1) nach UTF-8.

utf8.to_latin1(*String [,Ersatz]***)**
: Konvertiert einen UTF-8 *String* nach Latin-1 (ISO-8859-1). Zeichen, die nicht konvertiert werden können, werden durch den *Ersatz*-String ersetzt, oder durch "\x1A" wenn kein *Ersatz*-String angegeben ist.

SIEHE AUCH

lua-akfavatar(1) **lua**(1) **lua-akfavatar-ref**(3) **akfavatar-graphic**(3) **akfavatar-term**(3)