NAME
akfavatar.utf8 − module for UTF-8 support in Lua-AKFAvatar

SYNOPSIS
local utf8 = require "akfavatar.utf8"

DESCRIPTION
This module defines functions for UTF-8 strings. Many functions are replacements to the functions in
Lua’s string library.

UTF-8 is a character encoding for Unicode. A character can be encoded with one to four bytes. That’s
why it needs special handling. Most functions in the default string library of Lua can only handle encod-
ings with a single byte per character.

utf8.len(string)
Counts the number of characters in an UTF-8 encoded string.
Note: Control characters and invisible characters are counted, too.

utf8.sub(string, startchar [,endchar])
Like string.sub, but for UTF-8 strings.
Returns a substring from startchar to endchar. If startchar or endchar are negative, then they are
counted from the end of the string.
So, utf8.sub(s, 1, 3) returns the first 3 characters, while utf8.sub(s, -3) returns the last 3 characters.

utf8.char(...)
Like string.char but accepts higher numbers and returns an UTF-8 encoded string.

utf8.codepoint(string)
Return the codepoint of the first character of the given string.
Returns nil on error (but that’s not a real validity check).

utf8.codepoints(string [,startchar [,endchar]])
Like string.byte.
Returns the unicode numbers of the characters from startchar to endchar.
If you only need the first character, use utf8.codepoint() instead.

utf8.characters(string)
Iterator for the characters of an UTF-8 string.
A character may be a single- or multi-byte string.
Use like this:
for c in utf8.characters(line) do print(utf8.codepoint(c)) end

utf8.reverse(string)
Reverses an UTF-8 encoded string.
Note: combining characters are still problematic.

utf8.rep(string, n)
Returns the string repeated n times. It’s simply an alias for string.rep().

utf8.underlined(string)
Returns the string underlined (overstrike technique).

utf8.bold(string)
Returns the string in boldface (overstrike technique).

utf8.bom
Byte Order Mark.
Not really needed for UTF8, but sometimes used as signature.

utf8.check_bom(string)
Check, if the string starts with a UTF8-BOM.
utf8.check(string)
    Check if the string is an UTF-8 string.
    It’s just for checking if it is UTF-8 or not, not a validity check.
    Note: plain ASCII is also valid UTF-8.

utf8.check_unicode(string)
    Checks text for unicode encodings.
    Returns either of "UTF-8", "UTF-16BE", UTF-16LE", "UTF-32BE", "UTF-32LE" or nil if it cannot be detected.

utf8.from_ncr(string)
    Replaces numeric character references (NCR) with UTF-8 characters.
    For example "&#8364;" (decimal) or "&#x20AC;" (hexadecimal) for the Euro currency sign.

utf8.to_nrc(string)
    Replaces non-ASCII characters with numeric character references. The result is a plain ASCII string, but encoded.

utf8.from_latin1(string)
    Converts a string from Latin-1 (ISO-8859-1) to UTF-8.

utf8.to_latin1(string [,replacement])
    Converts an UTF-8 string into Latin-1 (ISO-8859-1). Characters which cannot be converted are replaced with the replacement string if given, or they are replaced with “\x1A”.

SEE ALSO
    lua-akfavatar(1) lua(1) lua-akfavatar-ref(3) akfavatar-graphic(3) akfavatar-term(3)