Inhaltsverzeichnis

Modul:Unicode convert

Usage

Converts Unicode character codes, always given in hexadecimal, to their UTF-8 or UTF-16 representation in upper-case hex or decimal. Can also reverse this for UTF-8. The UTF-16 form will accept and pass through unpaired surrogates e.g. `{{#invoke:Unicode convert|getUTF8|D835}}` → D835. The reverse function `fromUTF8` accepts multiple characters, and can have both input and output set to decimal.

When using from another module, you may call these functions as e.g. `unicodeConvert.getUTF8 { args = {'1F345'} }`, without a proper `frame` object.

To find the character code of a given symbol (in decimal), use e.g. Vorlage:Ml → 128049.

Code	Output
Vorlage:Mlx	F0 9F 8D 85
Vorlage:Mlx	240 159 141 133
Vorlage:Mlx	1F345
Vorlage:Mlx	127813
Vorlage:Mlx	D83C DF45
Vorlage:Mlx	55356 57157

See also

Vorlage:Unicode templates

```
local p = {}

-- NOTE: all these functions use frame solely for its args member.
-- Modules using them may therefore call them with a fake frame table
-- containing only args.

p.getUTF8 = function (frame)
        local ch = mw.ustring.char(tonumber(frame.args[1] or '0', 16) or 0)
        local bytes = {mw.ustring.byte(ch, 1, -1)}
        local format = ({
                ['10'] = '%d',
                dec = '%d'
        })[frame.args['base']] or '%02X'
        for i = 1, #bytes do
                bytes[i] = format:format(bytes[i])
        end
        return table.concat(bytes, ' ')
end

p.getUTF16 = function (frame)
```

```
        local codepoint = tonumber(frame.args[1] or '0', 16) or 0
        local format = ({ -- TODO reduce the number of options.
                ['10'] = '%d',
                dec = '%d'
        })[frame.args['base']] or '%04X'
        if codepoint <= 0xFFFF then -- NB this also returns lone surrogate charac
                return format:format(codepoint)
        elseif codepoint > 0x10FFFF then -- There are no codepoints above this
                return ''
        end
        codepoint = codepoint - 0x10000
        bit32 = require('bit32')
        return (format .. ' ' .. format):format(
                bit32.rshift(codepoint, 10) + 0xD800,
                bit32.band(codepoint, 0x3FF) + 0xDC00)
end

p.fromUTF8 = function(frame)
        local basein = frame.args['basein'] == 'dec' and 10 or 16
        local format = frame.args['base'] == 'dec' and '%d ' or '%02X '
        local bytes = {}
        for byte in mw.text.gsplit(frame.args[1], '%s') do
                table.insert(bytes, tonumber(byte, basein))
        end
        local chars = {mw.ustring.codepoint(string.char(unpack(bytes)), 1, -1)}
        return format:rep(#chars):sub(1, -2):format(unpack(chars))
end

return p
```