

Modul:Unichar/Doku

Ausgabe: 17.04.2026

Letzte Änderung: 23.02.2022

Seite von

Inhaltsverzeichnis

- [1. Modul:Unichar/Doku](#)
- [2. Modul:Unichar](#)

Modul:Unichar/Doku

Dies ist die Dokumentationsseite für [Modul:Unichar](#)

Tests

- U+00A9 © <reserved-00A9>
- U+0378 <reserved-0378>
- U+AC00 HANGUL SYLLABLE GA
- U+0020 <reserved-0020>

Modul:Unichar

Tests

- U+00A9 © <reserved-00A9>
- U+0378 <reserved-0378>
- U+AC00 HANGUL SYLLABLE GA
- U+0020 <reserved-0020>

```
local p = {}

local Unicode_data = require "Module:Unicode data"

local function errorf(level, ...)
    if type(level) == "number" then
        return error(string.format(...), level + 1)
```

```

else -- level is actually the format string.
    return error(string.format(level, ...), 2)
end
end

-- from [[Template:Unichar]]
local styles = {
    smallcaps = 'class="smallcaps" style="font-variant: small-caps; font-size: smaller
    monospace = 'style="font-family: monospace, monospace;"', -- why repeat it?
    background_color = 'style="background: lightblue;"', -- for space characters
}
local function style(text, type)
    if not styles[type] then
        errorf("Style %s not recognized", type)
    end
    return ('<span %s>%s</span>'):format(styles[type], text)
end

local U = mw.ustring.char
local function show(codepoint)
    -- Format characters that at least are visible to me.
    -- The parentheses will short-circuit the evaluation of some of the conditions.
    -- Arabic number sign, Arabic sign sanah, Arabic footnote marker, Arabic sign safh:
    if 0x600 <= codepoint and (codepoint <= 0x604
        -- Arabic end of ayah, Syriac abbreviation mark, Arabic disputed e
        or codepoint == 0x6DD or codepoint == 0x70F or codepoint == 0x8E2
        -- interlinear annotation anchor, separator, terminator
        or 0xFFFF9 <= codepoint and (codepoint <= 0xFFFFB
        -- shorthand format letter overlap, continuing overlap, downstep
        or 0x1BCA0 <= codepoint and codepoint <= 0x1BCA3))
        or Unicode_data.is_printable(codepoint) then
        local printed_codepoint = U(codepoint)
        if mw.ustring.toNFC(printed_codepoint) ~= printed_codepoint then
            -- Prevent MediaWiki software from normalizing the character.
            printed_codepoint = ("%#x%X;"):format(codepoint)
        end
        if Unicode_data.is_combining(codepoint) then
            printed_codepoint = "" .. printed_codepoint
        end
        if Unicode_data.is_whitespace(codepoint) then
            printed_codepoint = style(printed_codepoint, "background_color")
        end
        return printed_codepoint
    else
        return ""
    end
end

local function u_plus(codepoint)
    return ("U+%04X"):format(codepoint)
end

local function get_codepoint(args, arg)
    local val = args[arg]
    local is_negative = false

    if type(val) ~= "string" then
        errorf("code point in [[hexadecimal]] expected, got %s", type(val))
    elseif val:find('^%s*%-') then
        -- Negative number strings yield a bizarre value:
        -- tonumber("-1", 16) -> 1.844674407371e+19.
        -- Strip initial minus.
        val = val:match("%-(.)")
        is_negative = true
    end

    local codepoint = tonumber(val, 16)

```

```

        or errorf("code point in [[hexadecimal]] expected, got %q", val)

    if is_negative then
        codepoint = -codepoint
    end

    if not (0 <= codepoint and codepoint <= 0x10FFFF) then
        errorf("code point %d out of range", codepoint)
    end

    return codepoint
end

function p.unichar(frame)
    local args = frame.args[1] and frame.args or frame:getParent().args
    local codepoint = get_codepoint(args, 1)

    local name_or_label = Unicode_data.lookup_name(codepoint)
    local is_label = name_or_label:sub(1, 1) == "<"

    return ("%s %s %s"):format(
        style(u_plus(codepoint), "monospace"),
        show(codepoint),
        is_label and name_or_label or style(name_or_label, "smallcaps"))
end

return p

```