



Inhaltsverzeichnis

Modul:DecodeEncode

Implements Lua functions `mw.text.decode`, `mw.text.encode` in a module.

```
{{#invoke:decodeEncode|decode|s=Source&nbsp;text}} → Source text
```

See [List of XML and HTML character entity references](#).

Inhaltsverzeichnis

1 Decode (© → ©)	2
1.1 Decode a reduced set only	3
2 Encode (© → ©)	3
2.1 character set to encode	3
3 Template	3
4 See also	4

Decode (© → ©)

Note 2021-09-26: Possible bug: Decoding   works, but   doesn't.

Decodes **Named Entities** *from* entity name *into* a regular (unicode) character:

`©` → ©

`>` → >

All welldefined named entities are decoded ([HTML Named character references](#), formally: as defined in the [PHP table](#)).

A regular, rendered sentence:

"At 100 °F, & with a "burning" sun above, we , we /walked/."

In code:

```
"At 100&nbsp;&deg;F, & with a &quot;burning&quot; sun above, we &frasl;
walked&frasl;." -- wikitext
```

Processing:

```
{{#invoke:decodeEncode|decode|s=At 100 °F, & with a "burning" sun above, we
walked.}} →
```

```
At 100 °F, & with a "burning" sun above, we walked. -- In code: straight
characters, no named entities.
```

Renders, again:

"At 100 °F, & with a "burning" sun above, we /walked/."

Decode a reduced set only

By setting **Vorlage:Para**, only these five entity names are decoded: '<', '>', '&', '"', ' '; (that is, into '<', '>', '&', '"', ' ').

Note: There is a difference with the relevant Lua parameter. (This only concerns your task if you also work directly with the Lua `mw.text.decode` function). Lua documentation defines parameter **Vorlage:Para**, having this effect: when *omitted or false*, only the reduced set of entities is recognized and decoded. This use of 'false' is *inverted* in using **Vorlage:Para: Vorlage:Para = Vorlage:Para**.

Also, this module ignores the "omitted" logic: **Vorlage:Para** should be set explicitly to 'true' to be effective.

Encode (© → ©)

Function `encode` encodes some entity-named characters into that name (for example: `&` → `&`).

Regular sentence:

"At >100 °F, & with a "burning" sun above, we walked. ©"

In code:

```
"At >100 °F, & with a "burning" sun above, we walked. ©"
```

Encode:

```
{{#invoke:decodeEncode|encode|s=At >100 °F, & with a "burning" sun above, we walked. ©|charset=&<>{{!}}°"'&©}}
```

→

```
At &gt;100 &#176;F, &amp; with a &quot;burning&quot; sun above, we walked. &#169;
```

Renders as:

"At >100 °F, & with a "burning" sun above, we walked. ©"

character set to encode

Per Lua documentation, only a small set of characters is processed. The charset can be set (expanded) by using **Vorlage:Para**.

Example: **Vorlage:Para** (the default), **Vorlage:Para**; characters not in the default will be replaced by their decimal entity: © → `©` (hexadecimal number, not decimal nor named `©`)

Template

NOTE: 2021-09-13: The encode function with user-supplied charset is now used productively in **Vorlage:TI and **Vorlage:TI**. Before implementing breaking changes here, these templates need to be adjusted accordingly!**

See also

- [mw.text.decode](#)
- [mw.text.encode](#)

Vorlage:Navbar wikitext-handling templates

- [Module:Urldecode](#)

```
local p = {}

function _getBoolean( boolean_str )
  -- from: module:String; adapted
  -- requires an explicit true
  local boolean_value

  if type( boolean_str ) == 'string' then
    boolean_str = boolean_str:lower()
    if boolean_str == 'true' or boolean_str == 'yes' or boolean_str == '1' then
      boolean_value = true
    else
      boolean_value = false
    end
  elseif type( boolean_str ) == 'boolean' then
    boolean_value = boolean_str
  else
    boolean_value = false
  end
  return boolean_value
end

function p.decode( frame )
  local s
  local subset_only

  s = frame.args['s'] or ''
  subset_only = _getBoolean(frame.args['subset_only'] or false)

  return p._decode( s, subset_only )
end

function p._decode( s, subset_only )
  local ret = nil;

  s = mw.ustring.gsub( s, '&thinsp;', '&ThinSpace;' ) -- Workaround for bug: &T

  ret = mw.text.decode( s, not subset_only )

  return ret
end

function p.encode( frame )
  local s
  local charset

  s = frame.args['s'] or ''
  charset = frame.args['charset']
```



```
        return p._encode( s, charset )
    end

    function p._encode( s, charset )
        -- example: charset = '_&@-°\\\"\'\'=\' -- do escape with backslash not %;
        local ret

        if charset ~= (nil or '') then
            ret = mw.text.encode( s, charset )
        else
            -- use default: charset = '<>&\"\' ' (outer quotes = lua required)
            ret = mw.text.encode( s )
        end

        return ret
    end

    return p
end

return p
```