bcp-47

main no status

Parse and stringify BCP 47 language tags.

Contents

What is this?

This is a package that can parse BCP 47 language tags to an object representing them, and serialize those objects back into language tags. It supports a forgiving mode to handle incorrect BCP 47 tags and can emit warnings about problems in incorrect tags.

When should I use this?

You can use this package if you need to access the data stored in BCP 47 language tags. You can also use this package if you want to check (lint) or manipulate tags.

Install

This package is ESM only. In Node.js (version 12.20+, 14.14+, or 16.0+), install with npm:

```
npm install bcp-47
```

In Deno with Skypack:

```
import * as bcp47 from 'https://
      cdn.skypack.dev/bcp-47@2?dts'
```

- `wooorm/iso-3166` — ISO 3166 codes
- `wooorm/iso-639-2` — ISO 639-2 codes
- `wooorm/iso-639-3` — ISO 639-3 codes
- `wooorm/iso-15924` — ISO 15924 codes
- `wooorm/un-m49` — UN M49 codes

Contribute

Yes please! See How to Contribute to Open Source.

License

Types

This package is fully typed with TypeScript. It exports additional `Schema`, `Extension`, `Warning`, and `Options` types that model their respective interfaces.

Compatibility

This package is at least compatible with all maintained versions of Node.js. As of now, that is Node.js 12.20+, 14.14+, and 16.0+. It also works in Deno and modern browsers.

Security

This package is safe.

Related

- `wooorm/bcp-47-match` — match BCP 47 language tags with language ranges per RFC 4647
- `wooorm/bcp-47-normalize` — normalize, canonicalize, and format BCP 47 tags

In browsers with Skypack:

```html
<script type="module">
  import * as bcp47 from 'https://
        cdn.skypack.dev/bcp-47@2?min'
</script>
```

Use

```js
import {parse, stringify} from 'bcp-47'

const schema = parse('hy-Latn-IT-
        arevela')

console.log(schema)
console.log(stringify(schema))
```

Yields:

```js
{ language: 'hy',
  extendedLanguageSubtags: [],
  script: 'Latn',
  region: 'IT',
  variants: ['arevela'],
  extensions: [],
  privateuse: [],
  irregular: null,
  regular: null }
'hy-Latn-IT-arevela'
```

API

This package exports the following identifiers: `parse` and `stringify`. There is no default export.

`parse(tag[, options])`

Parse a BCP 47 tag into a language schema. Note that the algorithm is case insensitive.

`options.normalize`

Whether to normalize legacy tags when possible (`boolean`, default: `true`). For example, `i-klingon` does not match the BCP 47 language algorithm but is considered valid by BCP 47 nonetheless. It is suggested to use `tlh` instead (the ISO 639-3 code for Klingon). When `normalize` is `true`, passing `i-klingon` or other deprecated tags, is handled as if their suggested valid tag was given instead.

`options.forgiving`

By default, when an error is encountered, an empty object is returned. When in forgiving mode, all found values up to the point of the error are included (`boolean`, default: `false`). So, for example, where by default `en-GB-abcdefghi` an empty object is returned (as the language variant is too long), in `forgiving`

`function warning(reason, code, offset)`

Called when an error occurs.

Parameters

- `reason` (`string`) — English reason for failure
- `code` (`number`) — code for failure
- `offset` (`number`) — index-based place where the error occurred in the tag

Warnings

code	reason
1	Too long variant, expected at most 8 characters
2	Too long extension, expected at most 8 characters
3	Too many extended language subtags, expected at most 3 subtags
4	Empty extension, extensions must have at least 2 characters of content
5	Too long private-use area, expected at most 8 characters
6	Found superfluous content after tag

- `no-nyn`
- `zh-guoyu`
- `zh-hakka`
- `zh-min`
- `zh-min-nan`
- `zh-xiang`

`schema.irregular`

One of the `irregular` tags (`string`): tags that are seen as invalid by the algorithm). Valid values are:

- `en-GB-oed`
- `i-ami`
- `i-bnn`
- `i-default`
- `i-enochian`
- `i-hak`
- `i-klingon`
- `i-lux`
- `i-mingo`
- `i-navajo`
- `i-pwn`
- `i-tao`
- `i-tay`
- `i-tsu`
- `sgn-BE-FR`
- `sgn-BE-NL`
- `sgn-CH-DE`

mode the `language` of `schema` is populated with `en` and the `region` is populated with `GB`.

`options.warning`

When given, `warning` is called when an error is encountered (`Function`).

Returns

Parsed BCP 47 language tag (`Schema`).

Throws

When `tag` is `null` or `undefined`.

`stringify(schema)`

Compile a `schema` to a BCP 47 language tag.

Returns

BCP 47 language tag (`string`).

`Schema`

A schema represents a language tag. A schema is deemed empty when it has neither `language`, `irregular`, `regular`, nor `privateuse` (where an empty `privateuse` array is handled as no `privateuse` as well).

`schema.language`

Two or three character ISO 639 language code, four character reserved language code, or 5 to 8 (inclusive) characters registered language subtag (`string`). For example, `en` (English) or `cmn` (Mandarin Chinese).

`schema.extendedLanguageSubtags`

Selected three-character ISO 639 codes(`Array<string>`), such as `yue` in `zh-yue-HK` (Chinese, Cantonese, as used in Hong Kong SAR).

`schema.script`

Four character ISO 15924 script code (`string`), such as `Latn` in `hy-Latn-IT-arevela` (Eastern Armenian written in Latin script, as used in Italy).

`schema.region`

Two alphabetical character ISO 3166-1 code or three digit UN M49 code (`string`). For example, `CN` in `cmn-Hans-CN` (Mandarin Chinese, Simplified script, as used in China) or `419` in `es-419` (Spanish as used in Latin America and the Caribbean).

`schema.variants`

5 to 8 (inclusive) character language variants (`Array<string>`), such as `rozaj` and `biske` in `sl-rozaj-biske` (San Giorgio dialect of Resian dialect of Slovenian).

`schema.extensions`

List of extensions (`Array<Object>`), each an object containing a one character `singleton`, and a list of `extensions` (`string`). `singleton` cannot be `x` (case insensitive) and `extensions` must be between two and eight (inclusive) characters. For example, an extension would be `u-co-phonebk` in `de-DE-u-co-phonebk` (German, as used in Germany, using German phonebook sort order), where `u` is the `singleton` and `co` and `phonebk` are its extensions.

`schema.privateuse`

List of private-use subtags (`Array<string>`), where each subtag must be between one and eight (inclusive) characters.

`schema.regular`

One of the `regular` tags (`string`): tags that are seen as something different by the algorithm. Valid values are:

- `art-lojban`
- `cel-gaulish`
- `no-bok`