Engineer

CLDR

言語と地域ごとの書式や並べ替えなどの慣習を記録したデータ群

どのようなものか

CLDRは、Unicodeとともに使うロケールデータの集まりです。Unicodeは文章を書く文字を定めます。一方、CLDRは、ある地域のある言語の読み手が、日付、数値、価格、単位、一覧をどのように表し、名前をどの順序に並べることを期待するかを記録します。どちらもUnicode Consortiumが管理しています。CLDRの現在の正式名称はUnicode Common Locale Data & Rulesです。以前はCommon Locale Data Repository、仕様はLocale Data Markup Language、略してLDMLと呼ばれており、仕様では、旧称による参照も引き続き有効としています。

プロジェクトのサイトは、三つの構成要素を挙げています。一つ目は、ソフトウェアで使えるよう整理・構造化したデータ自体です。二つ目は、Unicode Technical Standard #35という仕様で、データの構造とアルゴリズムによる使い方を、適合性の要件とともに記述しています。三つ目は、言語の専門家からデータを集め、妥当性と整合性を検証し、開発者が使う形式へ変換するコードです。データの範囲を知るには、仕様自身の要約が役立ちます。ロケール、単位、数字体系、暦の識別子、日付、時刻、タイムゾーン、数値、通貨、測定値の書式、そして並べ替えと検索です。複数形の規則、一覧の書式パターン、言語や地域の名前も含まれます。

データはロケール別に整理されています。ロケールは言語を表し、文字体系と地域でさらに絞り込めます。各ロケールは親と異なる内容だけを保持します。仕様は英語を例に挙げ、`en`がデータの大部分を持ち、アイルランドで使う英語の`en_IE`は通貨などのわずかな項目だけを持つと説明しています。各言語の上にはrootというロケールがあり、その言語の主要な地域のロケールは、必要なものを親がすべて持つため、独自のデータを持ちません。通貨、単位系、週の慣習といった一部の情報は、言語に関係なく地域だけに属します。したがって、米国で使うフランス語のデータ一式がなくても、要求すれば書式は`fr`から取得し、金額は米ドルで表します。

元のデータは、仕様で定めるXML形式でリリースごとに公開され、JSONとPOSIX形式へ変換するツールも提供されています。リリースは年二回のサイクルに従い、CLDR 49は2026年10月22日に公開されました。新しいデータはオンラインのSurvey Toolから取り込みます。通常は年一回開かれ、新しく追加した構造のデータ収集と、以前のリリースのデータ修正を行います。取り込む内容は、母語話者と言語の専門家が検証します。

概念を判断に生かす

「CLDRとは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 日付、通貨、名前の並び順まで含め、販売先のそれぞれの市場で自然に読めるソフトウェアにします。

サービスを見る お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 英語・デンマーク語のi18nシステムを構築 — 12,027 英語・デンマーク語の言語ごとのメッセージ

英語・デンマーク語のi18nシステムを構築 お問い合わせ →

なぜ重要なのか

プロジェクトのサイトは、問題を率直に述べています。日付、数値、金額、測定値を正しく書くことは、「言語や地域をまたぐと、多くの人が認識しているよりはるかに複雑」です。CLDRは、その複雑さをチームが言語ごとに学び直さずに済むようにするためのものです。100を超える言語を支え、データは母語話者と言語の専門家が検証しています。一つの製品チームが、新たに進出した市場の日付書式について行える以上のレビューを受けているのです。

多くのチームがCLDRを直接扱わないのは、意図した設計です。プロジェクトは、Java、C#、.NET、Swift、JavaScriptなどに加え、現在のすべてのオペレーティングシステムとブラウザーがデータを利用するとしています。利用は間接的なこともよくあります。アプリケーションがOSのサービスへ日付の書式化を依頼し、そのサービスが、Unicodeが提供するC、C++、Java、Rust用の実運用ライブラリーICUを呼び、ICUがCLDRを読みます。Node.jsは `Intl` オブジェクトをICU上に構築し、既定でICUの全データを同梱します。Pythonプログラムは Babel、RubyプログラムはTwitterCLDRを通じて同じデータへアクセスします。チームがロケール対応の書式化機能を呼ぶと、答えはほぼ常にCLDRから来ます。チーム自身が決めるのは、どのロケールを要求し、どのリリースを使うかです。

代わりの方法は、各言語の十二か月の名前、小数点の記号、桁区切り、アルファベットの順序を手作業で表にすることです。その表は、良くても書いた日に正しいだけで、以後は固定されたままです。一方、CLDRのSurvey Toolは毎年再開

し、公開済みデータを修正します。しかも、その表を書く人は、多くの場合、含まれる言語の一部しか話せません。以下に示す、同じ六つの地名をデンマーク語とスウェーデン語で並べる順序は、それらの言語を知らない開発者が推測できるものではありません。

実際の活用例

実行環境に含まれるCLDRのリリースを調べ、そのデータで、デンマークとスウェーデンの六つの地名を並べ替え、デンマークの三都市名を一覧につなぎ、四つのロケールの週の情報を取得します。

```
console.log("CLDR", process.versions.cldr);

const towns = ["Ørsted", "Odense", "Åsa", "Aabenraa", "Ærø", "Ängelholm"];
console.log("code point", [...towns].sort().join(", "));
for (const tag of ["en", "da", "sv"]) {
  const sorted = [...towns].sort(new Intl.Collator(tag).compare);
  console.log(tag.padEnd(10), sorted.join(", "));
}

const three = ["Aarhus", "Odense", "Aalborg"];
for (const tag of ["en-GB", "en-US", "da", "uk"]) {
  console.log(tag.padEnd(10), new Intl.ListFormat(tag).format(three));
}

for (const tag of ["en", "en-GB", "en-DK", "da"]) {
  const locale = new Intl.Locale(tag);
  const { firstDay } = locale.getWeekInfo();
  console.log(tag.padEnd(10), locale.maximize().toString(), firstDay);
}
```

Node 25.2.1では、次のように出力されます。

```
CLDR 47.0
code point Aabenraa, Odense, Ängelholm, Åsa, Ærø, Ørsted
en         Aabenraa, Ærø, Ängelholm, Åsa, Odense, Ørsted
da         Odense, Ängelholm, Ærø, Ørsted, Aabenraa, Åsa
sv         Aabenraa, Odense, Åsa, Ängelholm, Ærø, Ørsted
en-GB      Aarhus, Odense and Aalborg
en-US      Aarhus, Odense, and Aalborg
da         Aarhus, Odense og Aalborg
uk         Aarhus, Odense i Aalborg
en         en-Latn-US 7
```

```
en-GB      en-Latn-GB 1
en-DK      en-Latn-DK 1
da         da-Latn-DK 1
```

最初の行は、この実行環境をビルドした際のリリースです。CLDR 49が公開済みでも、47.0です。実行環境が持つのは、自分のビルド時のデータであって、存在する中で最新のデータではありません。

比較関数を渡さない単純な `sort()` は、Ä、Å、Æ、Øを、記号の付かないすべての文字より後へ、文字コード順に置きます。この順序は、三言語のどれにも一致しません。英語は各名前を基になる文字で並べるため、Aで始まる四つが先に来て、ØrstedはOdenseの後になります。デンマーク語のアルファベットの末尾はÆ、Ø、Åです。ÄはÆの変種として扱い、古いaa表記のAabenraaはÅで始まる名前の中に並べます。仕様の照合順序の部分は、これをデンマーク語の縮約「aa」と呼んでいます。スウェーデン語の末尾はÅ、Ä、Öで、ÆをÄの変種、ØをÖの変種として扱い、Aabenraaを二つのaで始まる名前として並べます。ノルウェー語ブークモールの `nb` は、デンマーク語と同じ順序を返します。どの規則もコードには書かれておらず、各言語のCLDRの照合規則から順序が決まります。

一覧も同じ仕組みです。各ロケールには、一覧の先頭、中間、末尾、二項目だけの一覧について、それぞれ別の部分を持つ書式パターンがあります。イギリス英語は「and」の前にコンマを置かず、アメリカ英語は置きます。デンマーク語は「og」、ウクライナ語は「i」でつなぎます。それらの言語の単語を一つも含まない、一回の呼び出しで実現します。

最後の四行は、言語ではなく地域に属するデータを示しています。各行には、タグ、CLDRの推定サブタグ（likely subtags）から `maximize()` が補完する完全な形式、週の最初の日を示すISO 8601の番号があり、1は月曜日、7は日曜日です。実行環境は、地域を指定しない `en` を米国で使うラテン文字の英語として解釈するため、週は日曜日から始まります。イギリスやデンマークで使う英語は、デンマーク語と同じく月曜日から始まります。言語に関係なく、週は地域で決まります。

CLDRが記録する慣習は、翻訳文を手作業で書く際にも正しく守る必要があります。このサイトのプロジェクトの一つ、言語をまたぐ数値のチェックは、英語の文字列からすべての数値、百分率、倍率、単位を取り出し、すべての翻訳にそれぞれが含まれるよう求めます。また、小数点、桁区切り、引用符など、各言語で使うべき慣習と使ってはいけない表記も確認します。

問題が起きるところ

書式化の結果を、固定の文字列と比較します。 `"1 500"` という固定文字列を、`new Intl.NumberFormat("sv").format(1500)` の結果として期待するテストは、出力が見た目どおりでも失敗します。スウェーデン語、ノルウェー語、ウクライナ語は、エディターで通常の空白に見えるU+00A0 NO-BREAK SPACEを桁区切りに使うためです。同じテストは、更新時にも壊れる場合があります。新しいCLDRリリースで、以前のデータが修正される可能性があるためです。書式化した出力は同じ書式化機能の出力と比較し、各部分を整数部、桁区切り、小数点、小数部として示す `formatToParts` で構造を確かめます。書式化した文字列を数値へ解析し直してはいけません。仕様は、言語に依存しないデータを保存・交換し、読み手に合わせて書式化することを勧めています。

二つの実行環境が、別々のリリースを持っています。 Node 25.2.1にはCLDR 47.0が含まれますが、プロジェクトは49を公開しています。ブラウザーが持つのは、自分のビルドに同梱された版です。サーバーとブラウザーで日付をそれぞれ書式化するページは、一つの値に二つの文字列を表示し、あるマシンで記録したスナップショットとの照合が、別のマシンでは失敗することがあります。各値の書式化を一か所で行い、テストのスナップショットとともに `process.versions.cldr` を記録し、実行環境を意識的に更新します。

存在しないロケールでも、エラーなしにフォールバックします。 `new Intl.NumberFormat("tlh")` は、実行環境にクリンゴン語のデータがなく、既定のロケールへ戻るため、例外を出さずにアメリカ英語として書式化します。Nodeの文書には、さらに気付きにくい例があります。small-icuオプションでビルドしたバイナリーは、通常、英語データだけを持つため、スペイン語の月名に「enero」ではなく「M01」や「January」を出力します。ロケールを信頼する前に、`resolvedOptions().locale` か `supportedLocalesOf` を確かめ、ICUの全データを含むビルドを使います。これはNodeの既定であり、公式バイナリーも使う構成です。

地域のないタグには、最もありそうな地域が補われます。 実行環境は `en` を `en-Latn-US` として読むため、アイルランド、イギリス、デンマークの読み手に `en` を渡すと、日曜日に始まる週と、月を先にした10/27/2026のような日付になります。その根拠となる推定サブタグのデータを、仕様は経験則から導いたものと説明し、将来変わり得るとしています。地域が分かる場合は言語とともに渡し、言語しか分からなければ、`maximize()` で、実行環境がどの地域を想定するかを確認します。

CLDRの識別子を、BCP 47のAPIへ渡します。 CLDRのファイルや例は、`en_IE` のようにアンダースコアを使い、仕様は `_` と `-` を同等と扱います。JavaScriptの `Intl` はそうではありません。`new Intl.NumberFormat("da_DK")` は、`RangeError: Invalid language tag: da_DK` を送出します。仕様自身が示すBCP 47への変換と同じように、境界で各 `_` を `-` へ置き換え、結果を `Intl.getCanonicalLocales` へ渡します。返る値は `da-DK` です。

参考資料

- Unicode CLDRプロジェクト：CLDRの概要、三つの構成要素、利用者と利用方法、リリース日程を説明します。
- UTS #35、第1部：コア：ロケール識別子、旧称、ロケール間の継承、推定サブタグ、データを言語に依存させないための助言を含む仕様です。
- UTS #35、第2部：一般：表示名、単位系、上で使った一覧の書式パターンを説明します。
- UTS #35、第5部：照合：デンマーク語を例に、CLDRが各言語に合わせて既定の並べ替え順序を調整する仕組みを説明します。
- Node.js：国際化対応：Nodeが `Intl` オブジェクトをICU上に構築する仕組み、ビルドの選択肢、バイナリーが何に対応するかを調べる方法を説明します。
- ECMAScript国際化API仕様：`Intl` の基礎となる標準です。上で使った `Locale` のメソッドと、その根拠となるCLDRの週のデータや推定サブタグも含みます。

用語集

実際の制約に合わせて選ぶ

業務の流れ、既存のインターフェース、変更を必要とする利用者をお知らせください。検証可能な納品内容と、お客様のチームが保守を続けられる引き継ぎ内容を定められます。

料金を見る お問い合わせ →

用語を実務に結び付ける

「CLDRとは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く

連絡先カード

心を込めて丁寧に制作： D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/cldr/