Engineer

CLDR

données Unicode de format et de tri par langue et région

Ce que c’est

CLDR est un ensemble de données de locale qui accompagne Unicode. Unicode définit les caractères d’un texte ; CLDR consigne les conventions attendues par les lecteurs d’une langue dans une région donnée pour écrire une date, un nombre, un prix, une unité ou une liste, ainsi que pour trier les noms. Unicode Consortium maintient les deux. Le nom complet actuel est Unicode Common Locale Data & Rules. Le projet s’appelait auparavant Common Locale Data Repository, et sa spécification portait le nom de Locale Data Markup Language, ou LDML. La spécification précise que les références sous les anciens noms restent valables.

Le site du projet distingue trois parties. La première est constituée des données elles-mêmes, sélectionnées et structurées pour être utilisées par des logiciels. La deuxième est la spécification, Unicode Technical Standard #35, qui décrit la structure des données et leur emploi par les algorithmes, avec les exigences de conformité. La troisième est le code qui recueille les données auprès de spécialistes des langues, vérifie leur validité et leur cohérence et les convertit dans les formats utilisés par les développeurs. Le résumé du périmètre des données dans la spécification offre un bon repère : identifiants des locales, unités, systèmes de numération et calendriers ; formats des dates, heures, fuseaux, nombres, monnaies et mesures ; tri et recherche. S’y ajoutent les règles de pluriel, les modèles de listes et les noms des langues et régions.

Les données sont organisées par locale : une langue, éventuellement précisée par une écriture et une région. Chaque locale ne contient que les différences avec sa parente. L’exemple de la spécification est l’anglais : `en` contient l’essentiel et `en_IE`, anglais d’Irlande, seulement quelques éléments comme la monnaie. Une locale nommée root se trouve au-dessus de toutes les langues. La locale de la région principale d’une langue ne porte aucune donnée propre, car sa parente détient déjà tout le nécessaire. Quelques éléments appartiennent

uniquement à la région, quelle que soit la langue : monnaie, système de mesure et conventions de semaine. Si aucun jeu de données propre n'existe pour le français des États-Unis, une demande de cette locale obtient donc ses formats de `fr` et ses montants en dollars américains.

Les données sources sont publiées à chaque version en XML selon le format de la spécification, avec des outils de conversion vers JSON et POSIX. Le calendrier prévoit deux cycles par an, et CLDR 49 est sorti le 22 octobre 2026. Les nouvelles données passent par un Survey Tool en ligne, généralement ouvert une fois par an, pour alimenter les nouvelles structures comme pour corriger les versions précédentes. Des locuteurs natifs et experts linguistiques valident les apports.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce que le CLDR? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. Un logiciel qui se lit naturellement sur chaque marché où vous vendez, y compris les dates, les devises et le classement des noms.

Découvrir nos prestations Contactez-nous →

Vérifiez les hypothèses derrière le terme

Appuyez-vous sur l'explication et les réalisations liées pour formuler les questions de votre projet. Nous pouvons examiner les hypothèses avec vous avant le choix d'un outil. Système i18n anglais/danois — 12,027 messages par langue, en anglais et en danois

Système i18n anglais/danois Contactez-nous →

Pourquoi c'est important

Le site du projet énonce clairement le problème : écrire correctement une date, un nombre, un montant ou une mesure est « bien plus compliqué d'une langue et d'une région à l'autre que la plupart des gens ne le pensent ». CLDR existe pour éviter à chaque équipe d'apprendre cette complexité langue par langue. Il prend en charge plus de 100 langues et ses données sont validées par des locuteurs natifs et experts linguistiques. C'est davantage de relecture qu'une équipe produit ne peut en consacrer au format des dates d'un marché qu'elle vient de rejoindre.

La plupart des équipes ne touchent jamais directement CLDR, et c'est voulu. Le projet compte parmi ses utilisateurs tous les systèmes d'exploitation et navigateurs modernes, ainsi que des langages comme Java, C#, .NET, Swift et JavaScript. L'utilisation est souvent indirecte : une application demande à un service du système de formater une date, ce service appelle ICU, bibliothèque Unicode destinée à un usage en production en C, C++, Java et Rust, et ICU lit CLDR. Node.js construit ses objets `Intl` sur ICU et fournit toutes les données ICU dans sa configuration standard. Les programmes Python accèdent aux

mêmes données par Babel, ceux en Ruby par TwitterCLDR. Lorsqu'une équipe appelle une fonction de formatage qui tient compte de la locale, la réponse vient presque toujours de CLDR. Il lui reste à décider quelle locale demander et quelle version exécuter.

L'alternative est une table écrite à la main : douze noms de mois par langue, un séparateur décimal, un séparateur de milliers et un ordre alphabétique. Au mieux, elle est correcte le jour de son écriture et reste figée ensuite, tandis que le Survey Tool de CLDR rouvre chaque année pour corriger les données déjà publiées. Elle est aussi généralement écrite par une personne ne parlant qu'une partie des langues couvertes. Un développeur qui ne connaît pas ces langues ne devinerait pas les ordres dans lesquels le danois et le suédois trient les six mêmes noms de lieux présentés ci-dessous.

À quoi cela ressemble en pratique

Demandez à l'environnement d'exécution quelle version de CLDR il contient. Laissez ensuite ces données trier six noms de lieux danois et suédois, assembler trois noms de villes danoises en liste et consulter les conventions de semaine de quatre locales :

```
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 affiche :

```
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
```

La première ligne est la version avec laquelle cet environnement a été construit. C'est 47.0, bien que CLDR 49 soit sorti : un environnement embarque les données de sa propre compilation, pas les plus récentes qui existent.

Un simple `sort()` sans comparateur place Ä, Å, Æ et Ø après toutes les lettres sans marque, dans l'ordre de leurs codes de caractères. Cela ne correspond à aucune des trois langues. L'anglais trie chaque nom selon sa lettre de base : les quatre noms commençant par un A viennent donc d'abord, et Ørsted suit Odense. L'alphabet danois se termine par Æ, Ø et Å. Le tri danois traite Ä comme une variante de Æ et range Aabenraa, écrit avec l'ancien double a, parmi les noms en Å. La partie de la spécification consacrée à la collation appelle cela la contraction danoise « aa ». L'alphabet suédois se termine par Å, Ä et Ö. Le tri suédois traite Æ comme une variante de Ä et Ø comme une variante de Ö, et trie Aabenraa comme un nom commençant par deux a. Le norvégien bokmål, `nb`, renvoie l'ordre danois. Rien de cela ne figure dans le code : chaque ordre vient des règles de collation CLDR de la langue.

Les listes fonctionnent de même. Chaque locale a un modèle avec des parties distinctes pour le début, le milieu et la fin d'une liste, ainsi que pour une liste de deux éléments. L'anglais britannique n'ajoute pas de virgule avant « and », l'anglais américain si. Le danois joint avec « og » et l'ukrainien avec « і », au moyen d'un seul appel qui ne contient aucun mot de ces langues.

Les quatre dernières lignes montrent des données appartenant à une région plutôt qu'à une langue. Chacune donne le tag, sa forme complète enrichie par `maximize()` à partir des sous-étiquettes probables de CLDR, et le premier jour de la semaine sous forme de nombre ISO 8601 : 1 pour lundi et 7 pour dimanche. L'environnement interprète `en` seul comme l'anglais en écriture latine des

États-Unis : sa semaine commence donc le dimanche. L'anglais de Grande-Bretagne ou du Danemark commence le lundi, comme le danois. La semaine dépend de la région, quelle que soit la langue.

Les conventions consignées par CLDR sont aussi celles qu'un texte traduit doit respecter à la main. Un projet de ce site, le contrôle des chiffres entre langues, extrait chaque nombre, pourcentage, multiplicateur et unité d'une chaîne anglaise et exige leur présence dans chaque traduction. Il vérifie aussi les conventions que chaque langue doit suivre ou éviter, dont séparateurs décimaux, groupements de milliers et guillemets.

Où les problèmes surviennent

Comparer le texte formaté à une chaîne fixe. Un test attendant `"1 500"` de `new Intl.NumberFormat("sv").format(1500)` échoue malgré une sortie visuellement identique. Le suédois, le norvégien et l'ukrainien séparent les milliers avec U+00A0 NO-BREAK SPACE, qu'un éditeur montre comme une espace ordinaire. Le même test peut casser à la mise à niveau : une nouvelle version de CLDR peut corriger les données d'une ancienne. Comparez la sortie formatée à celle du même formateur, vérifiez sa structure avec `formatToParts`, qui étiquette chaque partie comme partie entière (integer), séparateur de groupe (group), séparateur décimal (decimal) ou partie fractionnaire (fraction), et ne reconvertissez jamais le texte formaté en nombre. La spécification recommande de stocker et d'échanger des données indépendantes de la langue, puis de les formater pour le lecteur.

Deux environnements contiennent deux versions. Node 25.2.1 embarque CLDR 47.0 alors que le projet a publié 49. Un navigateur contient ce que sa propre compilation a livré. Une page formatant une date sur le serveur puis dans le navigateur peut montrer deux chaînes pour une valeur, et un instantané enregistré sur une machine peut échouer sur une autre. Formatez chaque valeur à un seul endroit, enregistrez `process.versions.cldr` avec les instantanés de test et mettez l'environnement d'exécution à niveau délibérément.

Une locale absente déclenche un repli sans erreur. `new Intl.NumberFormat("tlh")` formate en anglais américain sans lever d'exception, car l'environnement n'a pas de données pour le klingon et retombe sur sa locale de repli. La documentation Node décrit un cas plus discret : un binaire compilé avec l'option small-icu ne contient généralement que les données anglaises. Le nom d'un mois espagnol s'affiche alors sous la forme « M01 » ou « January » au lieu de « enero ». Vérifiez

`resolvedOptions().locale` ou `supportedLocalesOf` avant de vous fier à une locale et utilisez des compilations avec toutes les données ICU, configuration standard de Node et de ses binaires officiels.

Un tag sans région reçoit la plus probable. L'environnement lit `en` comme `en-Latn-US`. Un lecteur en Irlande, en Grande-Bretagne ou au Danemark recevant `en` obtient donc une semaine débutant le dimanche et des dates comme 10/27/2026, mois en premier. La spécification dit que les données de sous-étiquettes probables à l'origine de ce choix sont dérivées d'heuristiques et peuvent évoluer. Transmettez la région du lecteur avec sa langue dès qu'elle est connue. Si seule la langue l'est, appelez `maximize()` pour voir la région supposée par l'environnement.

Passer les identifiants CLDR à une API BCP 47. Les fichiers et exemples CLDR écrivent les locales avec des traits de soulignement, comme `en_IE`, et la spécification traite `_` et `-` comme équivalents. L'API `Intl` de JavaScript ne les considère pas comme équivalents : `new Intl.NumberFormat("da_DK")` lève `RangeError: Invalid language tag: da_DK`. Convertissez les identifiants au passage vers l'API, comme le prévoit la conversion BCP 47 de la spécification, en remplaçant chaque `_` par `-`. Passez ensuite le résultat à `Intl.getCanonicalLocales`, qui renvoie `da-DK`.

Pour aller plus loin

- Projet Unicode CLDR — ce qu'est CLDR, ses trois composants, qui l'utilise et comment, ainsi que le calendrier de publication.
- UTS #35, partie 1 : noyau — la spécification : identifiants de locale, anciens noms, héritage entre locales, sous-étiquettes probables et conseil de garder les données indépendantes de la langue.
- UTS #35, partie 2 : généralités — noms d'affichage, systèmes de mesure et modèles de listes employés ci-dessus.
- UTS #35, partie 5 : collation — adaptation par CLDR de l'ordre de tri de référence à chaque langue, avec le danois comme exemple.
- Node.js : prise en charge de l'internationalisation — construction des objets `Intl` sur ICU, options de compilation et détection des capacités d'un binaire.
- Spécification ECMAScript de l'API d'internationalisation — la norme derrière `Intl`, dont les méthodes de `Locale` utilisées plus haut et les données CLDR de semaine et de sous-étiquettes probables dont elles dépendent.

Glossaire

Choisissez selon vos contraintes réelles

Présentez le processus de travail, les interfaces existantes et les utilisateurs qui ont besoin du changement. Nous pouvons définir une livraison testable et une transmission permettant à votre équipe d'assurer la maintenance des éléments livrés.

Consulter les tarifs Contactez-nous →

Rendez la terminologie concrète

Demandez comment « Qu'est-ce que le CLDR ? » s'applique à votre organisation. Un exemple concret tiré de votre système actuel suffit pour engager la discussion.

Découvrir nos prestations

welcome@engineer.company +45 50 10 61 84 Copenhague, Danemark

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

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

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://engineer.company/fr/glossary/cldr/