Engineer

CLDR

Unicodes data om hvert språks datoer, tall og sortering

Hva det er

CLDR er en samling data om språk og regioner som utfyller Unicode. Unicode definerer tegnene en tekst skrives med; CLDR beskriver hvordan lesere av et bestemt språk i en bestemt region forventer å se en dato, et tall, en pris, en enhet eller en liste skrevet, og i hvilken rekkefølge de forventer at navn blir sortert. Unicode-konsortiet vedlikeholder begge deler. CLDRs fulle navn er i dag Unicode Common Locale Data & Rules. Tidligere het det Common Locale Data Repository, og spesifikasjonen het Locale Data Markup Language, forkortet LDML; ifølge spesifikasjonen er henvisninger under de gamle navnene fortsatt gyldige.

Prosjektets nettsted nevner tre deler. Den første er selve dataene, valgt ut og strukturert slik at programvare kan bruke dem. Den andre er spesifikasjonen, Unicode Technical Standard #35, som beskriver hvordan dataene er bygd opp og hvordan algoritmer bruker dem, med krav til samsvar. Den tredje er koden som samler inn dataene fra språkspesialister, kontrollerer at de er gyldige og konsistente, og konverterer dem til formatene utviklere bruker. Spesifikasjonens eget sammendrag av hva dataene dekker, er et godt kart: identifikatorer for locales, måleenheter, tallsystemer og kalendere; formatering av datoer, klokkeslett, tidssoner, tall, valutaer og måleverdier; og sortering og søk. Flertallsregler, listemønstre og navn på språk og regioner hører også med.

Dataene er ordnet etter locale: et språk, som kan avgrenses nærmere med et skriftsystem og en region. Hver locale inneholder bare det som skiller den fra den overordnede localen. Spesifikasjonens eksempel er engelsk: `en` inneholder det meste av dataene, og `en_IE`, engelsk slik det brukes i Irland, inneholder bare noen få ting, for eksempel valutaen. Øverst over alle språk ligger en locale som heter root, og localen for et språks primære region har ingen egne data, fordi den overordnede localen allerede inneholder alt den trenger. Noen få opplysninger hører til regionen alene, uansett språk: valutaen, målesystemet og

konvensjonene for uken. Har CLDR ingen egne data for fransk slik det brukes i USA, får man derfor formatene fra `fr` og beløpene i amerikanske dollar når man ber om akkurat den kombinasjonen.

Kildedataene publiseres for hver versjon som XML i formatet spesifikasjonen definerer, med verktøy som konverterer dem til JSON- og POSIX-formater. Utgivelsene følger en plan med to sykluser i året, og CLDR 49 kom ut 22. oktober 2026. Nye data kommer inn gjennom et nettbasert verktøy, Survey Tool, som vanligvis er åpent én gang i året for å samle inn data til nye strukturer og rette data som allerede er utgitt, og morsmålsbrukere og språkeksperter går gjennom det som kommer inn.

Bruk begrepet i en beslutning

Bruk «Hva er CLDR?» i en konkret beslutning. Vi kan knytte begrepet til systemet dere bygger eller kjøper. Programvare som leses naturlig i alle markeder du selger i, også datoene, valutaene og måten navn sorteres på.

Se tjenestene våre Ta kontakt →

Undersøk antakelsene bak ordet

Bruk forklaringen og det tilknyttede arbeidet til å formulere spørsmål for prosjektet. Vi kan undersøke antakelsene sammen før et verktøy velges. Bygde et engelsk/dansk i18n-system — 12,027 meldinger per språkversjon, på engelsk og dansk

Bygde et engelsk/dansk i18n-system Ta kontakt →

Hvorfor det er viktig

Prosjektets nettsted beskriver problemet rett ut: å formatere datoer, tall, valutaer og måleenheter for ulike språk og regioner er langt mer komplisert enn folk flest er klar over. CLDR finnes for at ikke hvert team skal måtte lære seg den kompleksiteten ett språk om gangen. Prosjektet dekker over 100 språk, og dataene blir gjennomgått av morsmålsbrukere og språkeksperter, en grundigere kontroll enn noe produktteam kan gi datoformatet i et marked det nettopp har gått inn i.

De fleste team rører aldri CLDR direkte, og det er meningen. Prosjektet regner alle moderne operativsystemer og nettlesere blant dem som bruker dataene, sammen med språk som Java, C#, .NET, Swift og JavaScript. Bruken er ofte indirekte: en applikasjon ber en tjeneste i operativsystemet om å formatere en dato, tjenesten kaller ICU, Unicodes produksjonsbibliotek for C, C++, Java og Rust, og ICU leser CLDR. Node.js bygger `Intl`-objektene sine på ICU og leverer som standard de fulle ICU-dataene; Python-programmer når de samme dataene gjennom Babel, og Ruby-programmer gjennom TwitterCLDR. Når et team kaller

en formateringsfunksjon som tar hensyn til locale, kommer svaret nesten alltid fra CLDR. Det teamet fortsatt bestemmer selv, er hvilken locale det ber om, og hvilken versjon det kjører.

Alternativet er en tabell skrevet for hånd: tolv månedsnavn per språk, et desimaltegn, et tusenskilletegn, en rekkefølge for alfabetet. I beste fall er en slik tabell riktig den dagen den blir skrevet, og deretter står den stille, mens CLDRs Survey Tool åpner hvert år for å rette data som allerede er utgitt. Den er som regel også skrevet av en som bare snakker enkelte av språkene i den. Hvordan svensk sorterer de seks stedsnavnene nedenfor, ville en norsk utvikler neppe gjette, og omvendt.

Slik ser det ut i praksis

Spør runtimen hvilken CLDR-versjon den har med seg, og la så dataene sortere seks danske og svenske stedsnavn, sette sammen tre danske bynavn til en liste og slå opp uken for fire 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 skriver ut:

```
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
```

Første linje er versjonen runtimen er bygd med. Den er 47.0, selv om CLDR 49 er kommet ut: en runtime har med seg dataene den ble bygd med, ikke de nyeste dataene som finnes.

En vanlig `sort()` uten sammenligningsfunksjon legger Ä, Å, Æ og Ø etter alle bokstavene fra A til Z, i rekkefølge etter tegnkodene deres, og det passer ikke med noen av de tre språkene. Engelsk sorterer hvert navn etter grunnbokstaven, så de fire navnene som begynner med A, kommer først, og Ørsted kommer etter Odense. Dansk avslutter alfabetet med Æ, Ø og Å, behandler Ä som en variant av Æ og sorterer Aabenraa, stavet med dobbel a etter eldre skikk, blant Å-navnene; spesifikasjonens del om sortering kaller dette den danske kontraksjonen «aa». Svensk avslutter alfabetet med Å, Ä og Ö, behandler Æ som en variant av Ä og Ø som en variant av Ö, og sorterer Aabenraa som et navn som begynner med to a-er. Norsk bokmål, `nb`, gir samme rekkefølge som dansk. Ingenting av dette står i koden; hver rekkefølge kommer fra CLDRs sorteringsregler for det aktuelle språket.

Listene fungerer på samme måte. Hver locale har et listemønster med egne deler for begynnelsen, midten og slutten av en liste, og for en liste på to. Britisk engelsk setter ikke komma foran «and», amerikansk engelsk gjør det, dansk binder sammen med «og» og ukrainsk med «і», og alt kommer fra ett kall som ikke inneholder et eneste ord på noen av disse språkene.

De fire siste linjene viser data som hører til en region og ikke til et språk. Hver linje gir språkkoden, den fulle formen som `maximize()` fyller ut med CLDRs mest sannsynlige skriftsystem og region, og ukens første dag som et tall etter ISO 8601, der 1 er mandag og 7 er søndag. Står det bare `en`, leser runtimen det som engelsk med latinsk skrift slik det brukes i USA, så uken begynner på søndag. Engelsk slik det brukes i Storbritannia eller i Danmark, begynner uken på mandag, slik dansk gjør: uken kommer fra regionen, uansett språk.

Konvensjonene CLDR registrerer, er også de en oversatt tekst må få riktig for hånd. Et av prosjektene på dette nettstedet, en tallkontroll på tvers av språk, trekker ut hvert tall, hver prosentsats, hver multiplikator og hver enhet fra en engelsk streng og krever at hver av dem finnes i hver oversettelse. Den kontrollerer også hvert språk mot konvensjonene det må bruke og ikke må bruke, blant annet desimalskilletegn, gruppering av tusener og anførselstegn.

Hvor det går galt

Formatert tekst sammenlignes som en fast streng. En test som forventer `"1 500"` fra `new Intl.NumberFormat("sv").format(1500)`, feiler selv om resultatet ser nøyaktig slik ut, fordi svensk, norsk og ukrainsk skiller tusener med U+00A0 NO-BREAK SPACE, som en editor viser som et vanlig mellomrom. Den samme testen kan også knekke ved en oppgradering, fordi en ny CLDR-versjon kan rette data som en eldre versjon leverte. Sammenlign formatert tekst med det den samme formatereren gir, sjekk strukturen med `formatToParts`, som merker hver del som integer, group, decimal eller fraction, og gjør aldri formatert tekst om til et tall igjen. Spesifikasjonens råd er å lagre og utveksle data i språknøytral form og formatere dem for leseren.

Server og nettleser har hver sin versjon. Node 25.2.1 leverer CLDR 47.0, mens prosjektet har kommet til 49, og en nettleser har med seg den versjonen den selv ble bygd med. En side som formaterer en dato på serveren og så igjen i nettleseren, kan vise to ulike strenger for én verdi, og et snapshot som er tatt opp på én maskin, kan feile på en annen. Formater hver verdi ett sted, lagre `process.versions.cldr` sammen med testenes snapshots, og oppgrader runtimen bevisst.

En manglende locale faller tilbake uten feil. `new Intl.NumberFormat("tlh")` formaterer som amerikansk engelsk uten å kaste en feil, fordi runtimen ikke har data for klingon og faller tilbake til standardlocalen sin. Dokumentasjonen til Node.js beskriver et mer lumskt tilfelle: en binærfil bygd med small-icu-alternativet har vanligvis bare engelske data, og der skrives et spansk månedsnavn ut som «M01» eller «January» i stedet for «enero». Sjekk `resolvedOptions().locale` eller `supportedLocalesOf` før du stoler på en locale, og bygg med de fulle ICU-dataene, som er standard i Node og det de offisielle binærfilene bruker.

En språkkode uten region får den mest sannsynlige. Runtimen leser `en` som `en-Latn-US`, så en leser i Irland, Storbritannia eller Danmark som får servert `en`, får en uke som begynner på søndag, og datoer skrevet som 10/27/2026, med måneden først. Dataene bak, som spesifikasjonen kaller likely subtags, er etter dens egen beskrivelse heuristisk utledet og kan endre seg over tid. Send med leserens region sammen med språket når den er kjent, og kall `maximize()` når bare språket er kjent, så du ser hvilken region runtimen kommer til å gå ut fra.

CLDRs identifikatorer sendes til et BCP 47-API. CLDRs filer og eksempler skriver locales med understrek, for eksempel `en_IE`, og spesifikasjonen behandler `_` og `-` som likeverdige. Det gjør ikke JavaScripts `Intl`: `new Intl.NumberFormat("da_DK")` kaster `RangeError: Invalid language tag: da_DK`. Konverter ved grensen slik spesifikasjonens egen konvertering til BCP 47 gjør, ved å bytte ut hver `_` med `-`, og send resultatet gjennom `Intl.getCanonicalLocales`, som returnerer `da-DK`.

Les mer

- Unicode CLDR Project — hva CLDR er, de tre delene, hvem som bruker det og hvordan, og utgivelsesplanen.
- UTS #35, Part 1: Core — spesifikasjonen: locale-identifikatorer, de tidligere navnene, arv mellom locales, likely subtags og rådet om å holde data språknøytrale.
- UTS #35, Part 2: General — visningsnavn, målesystemer og listemønstrene som brukes ovenfor.
- UTS #35, Part 5: Collation — hvordan CLDR tilpasser standardsorteringen til hvert språk, med dansk som eksempel.
- Node.js: Internationalization support — hvordan Node bygger `Intl`-objektene sine på ICU, byggealternativene og hvordan man finner ut hva en binærfil støtter.
- ECMAScript Internationalization API Specification — standarden bak `Intl`, blant annet `Locale`-metodene som brukes ovenfor, og CLDRs ukedata og likely subtags som de bygger på.

Ordliste

Velg ut fra deres faktiske rammer

Beskriv arbeidsflyten, eksisterende grensesnitt og brukerne som trenger endringen. Vi kan definere en testbar leveranse og en overlevering som gjør teamet i stand til å vedlikeholde løsningen.

Se hva det koster Ta kontakt →

Gjør begrepene praktiske

Spør hvordan «Hva er CLDR?» gjelder i organisasjonen deres. Et konkret eksempel fra dagens system er nok til å starte samtalen.

Se tjenestene våre

welcome@engineer.company +45 50 10 61 84 København, Danmark

Ta med deg denne siden

Visittkort

Laget med kjærlighet av D.

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

Opphavsrett © 2025 – nå · Engineer ApS · Alle rettigheter forbeholdt.

Skrevet ut fra https://engineer.company/nb/glossary/cldr/