

ICU4X Workshop

Unicode Technology Workshop 2023

Who We Are



Shane Carr
Chair of the Unicode
ICU4X TC



Robert Bastian
Unicode ICU4X TC

Today's Agenda

1. The i18n Stack
2. Challenges in the i18n space that ICU4X seeks to solve
3. Overview of the ICU4X project
4. **Interactive:** Build your first app using ICU4X
5. How the ICU4X locale data pipeline works
6. **Interactive:** Plug custom data into your app
7. Wrap up

What is i18n?

INTERNATIONALIZATION

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

I18N

Examples of i18n Operations

Today's date in various locales and calendars:

- en: 4/27/2023
- de-CH: 27.4.2023
- he: 6 5783 באייר
- bn: ৭/১০/১৪৪৪ যুগ
- zh: 2023年三月8
- ja: R5/4/27

Breakpoints in Japanese text (words may be multiple ideographs wide):

中|ワ|況|写|イノナ|開|億|が|や|へ|
者|43|俳|寺|式|7|沢|暮|ル|材|年|
る|あん|移|酔|むぎ|え|す|写|情|主|
逃|69|引|ぎ|う|ぱ|何|81|昇|フネイ
マ|本|因|ぱ|不|真|え|断|候|ら|。
吉|群|コクチ|賀|伝|ツエ|別|写|ユリ
キ|施|見|ら|力|1|取|目|れ|づ|ぱ|ぞ
|然|野|テ|倍|展|ぴ|る|て|け|京|答|
週|ネコ|入|整|料|勉|ら|げ|。

The i18n Stack

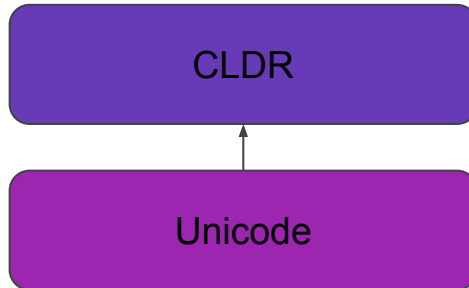
Unicode:
Foundational algorithms, specification,
character set



Unicode

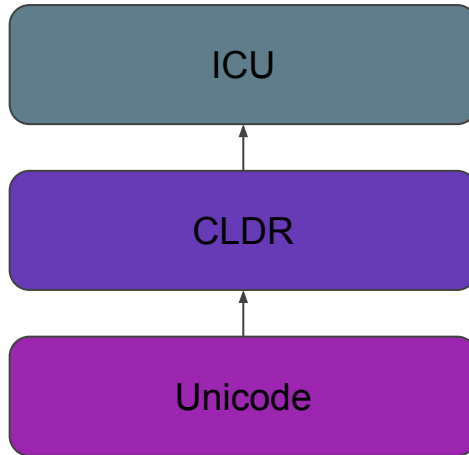
The i18n Stack

CLDR:
Data for hundreds of locales, specs for
MessageFormat, person names,
keyboards, ...



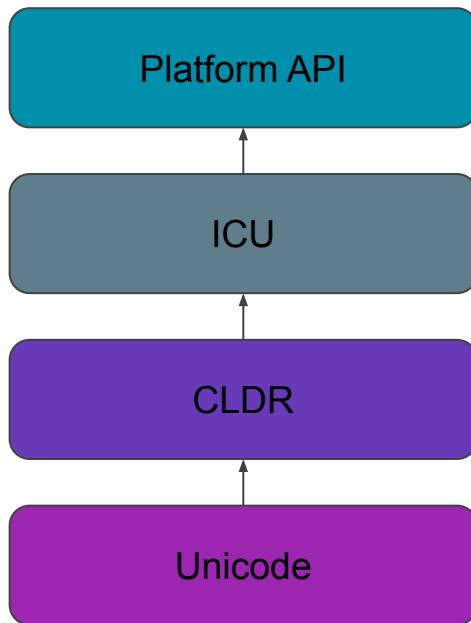
The i18n Stack

ICU:
Code that implements the algorithms in
Unicode with the data in CLDR

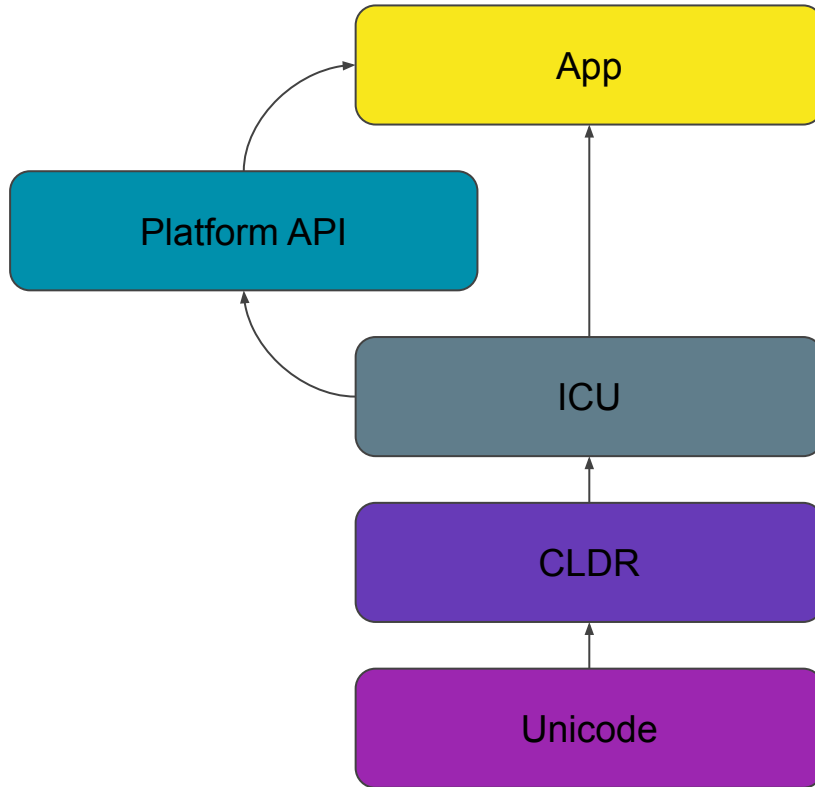


The i18n Stack

Platform API:
Making i18n easy to use from applications
(example: ECMA-402, android.icu)

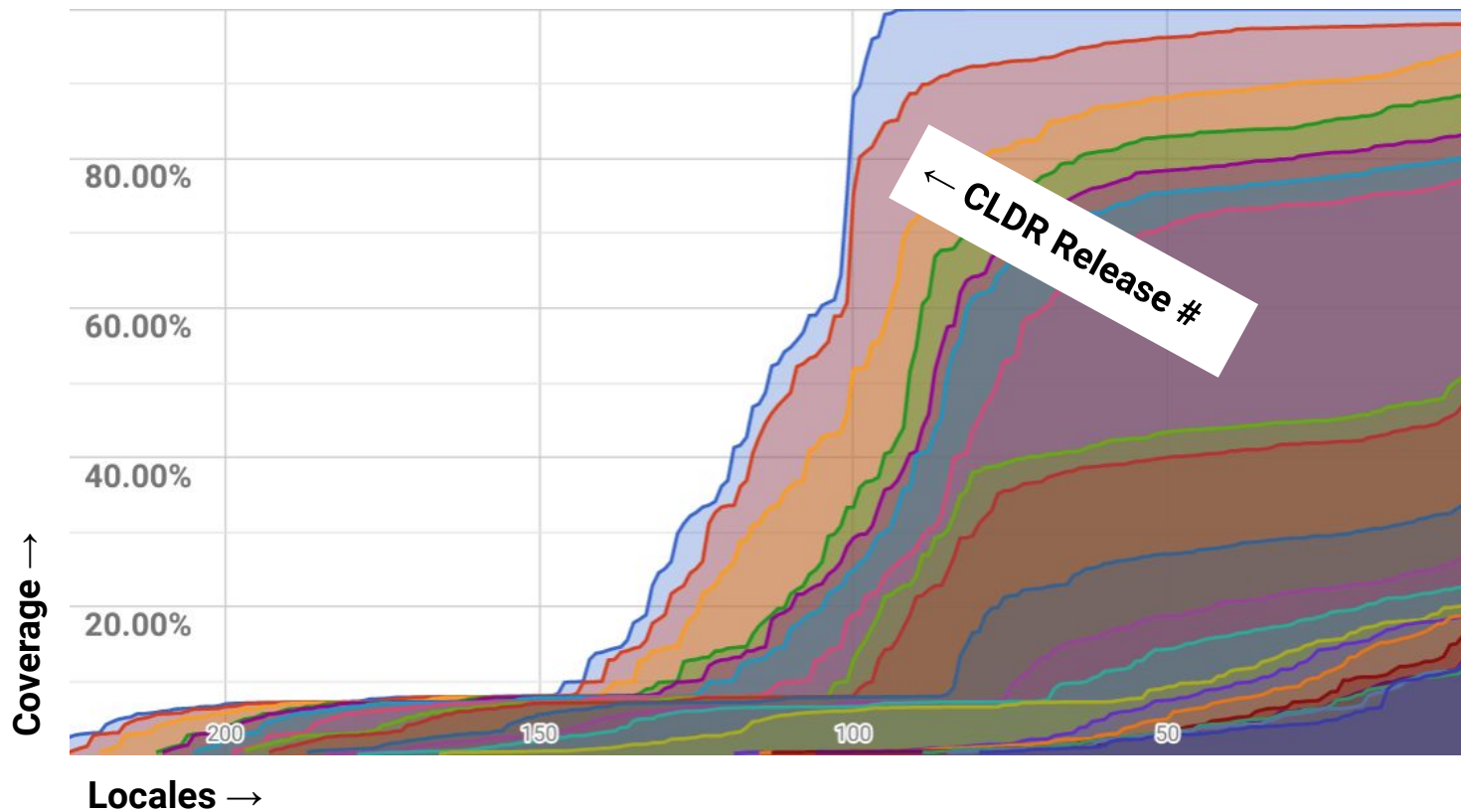


The i18n Stack



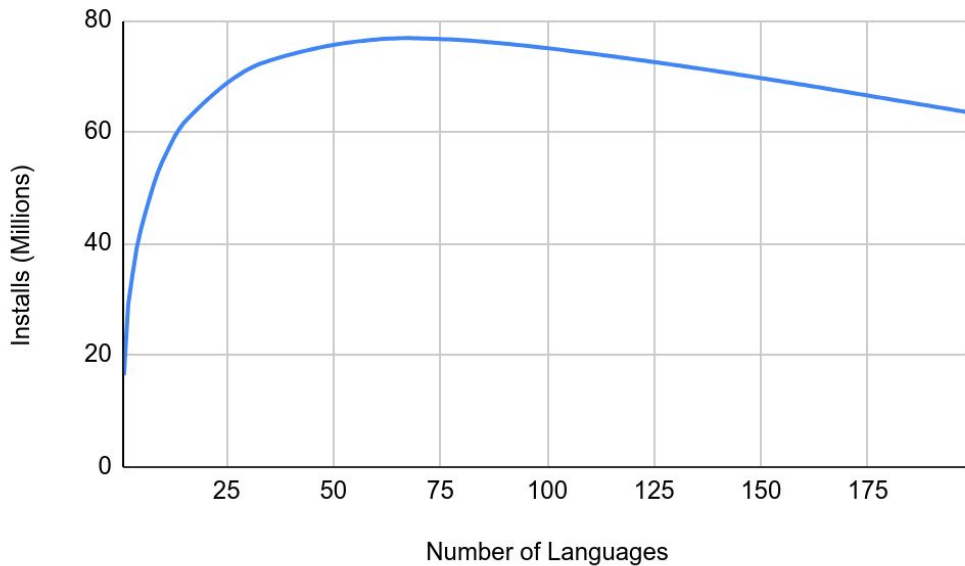
Challenges with Scaling i18n

Quadratic Growth of Data



Adding more default languages has diminishing returns

Cumulative Installs by # of Default Languages



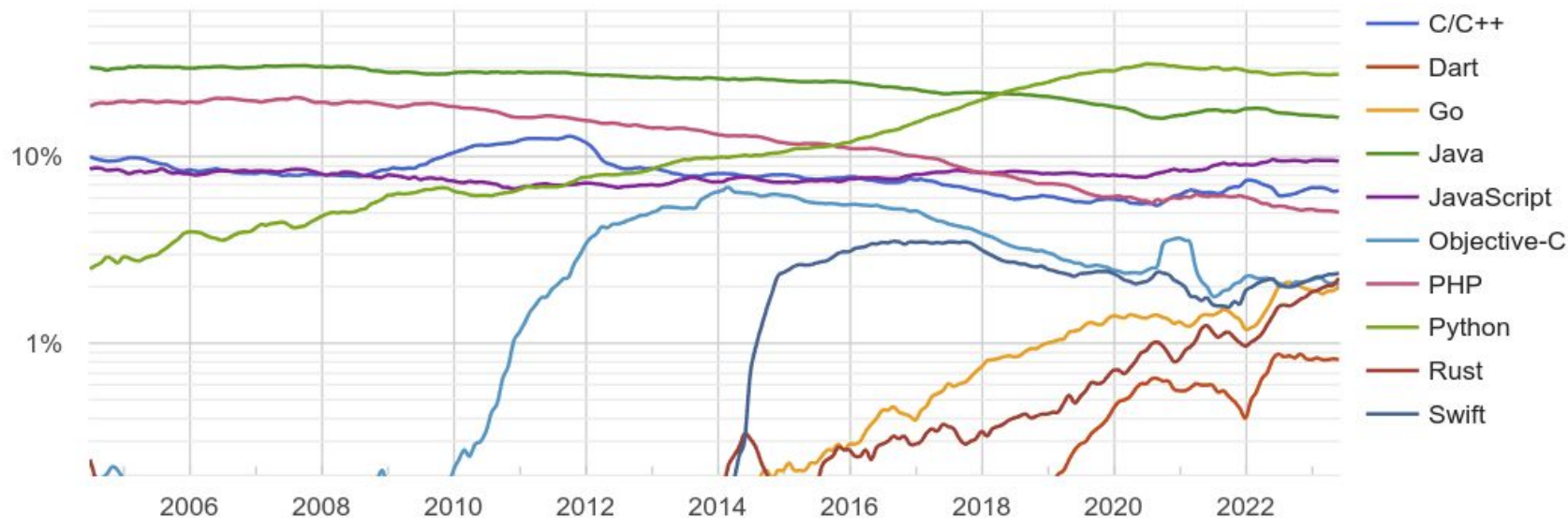
*If adding a language increases app size by 100 kB,
reducing total downloads by 0.25%*

Devices have different requirements



Hot new programming languages every year

PYPL Popularity of Programming Language



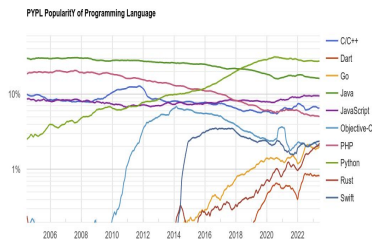
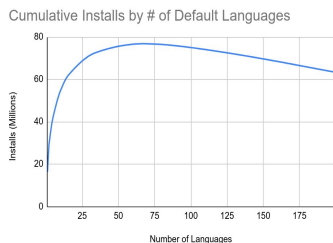
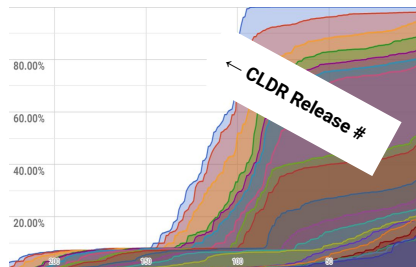
i18n-as-a-service doesn't make sense

- Low latency requirements
- Data-heavy algorithms
- Privacy implications

中|ワ|況|写|イノナ|開|億|が|や|へ|
者|43|俳|寺|式|7|沢|暮|ル|材|年|
る|あん|移|酔|むぎ|え|す|写|情|主|
逃|69|引|ぎ|う|ぱ|何|81|昇|フネイ
マ|本|因|ぱ|不|真|え|断|候|ら|。
吉|群|コクチ|賀|伝|ツエ|別|写|ユリ
キ|施|見|ら|カ|1|取|目|れ|づ|ぱ|ぞ
|然|野|テ|倍|展|ぴ|る|て|け|京|答|
週|ネコ|入|整|料|勉|ら|げ|。

Wish List for a Scalable i18n Library

- Pay for what you use and not what you don't use
- Add extra locales on demand
- Run on all types of devices
- Work in both today's and tomorrow's programming languages
- Run everything on-device



中|ワ|況|写|イ|ノ|ナ|開|億|が|や|
へ|者|43|俳|寺|式|7|沢|暮|ル|
材|年|る|あん|移|酔|む|ぎ|え|す|
写|情|主|逃|69|引|ぎ|う|ば|何|
81	昇	フ	ネ	イ	マ	本	因	ば	不	真
え	断	候	ら	。	吉	群	コ	ク	チ	賀
伝	ツ	エ	別	写	ユ	リ	キ	施	見	ら
カ|1|取|目|れ|づ|ば|ぞ|然|野|
テ|倍|展|び|る|て|け|京|答|週|
ネコ|入|整|料|勉|ら|げ|。

Enter ICU4X

What is ICU4X?

ICU4X is an internationalization library that is:

1. Portable by design

- Runs on all types of devices with supported wrappers for multiple programming languages

2. Lightweight

- Modular design, great for compile-time dead-code elimination and data slicing

3. Secure

- Written in Rust, a memory-safe language

Who are we?



ICU4X WebAssembly Demo

Fixed Decimal Formatting

Date Time Formatting

Segmenter

Locale

en

bn

other

Locale ID

Grouping Strategy

Auto

Never

Always

Min2

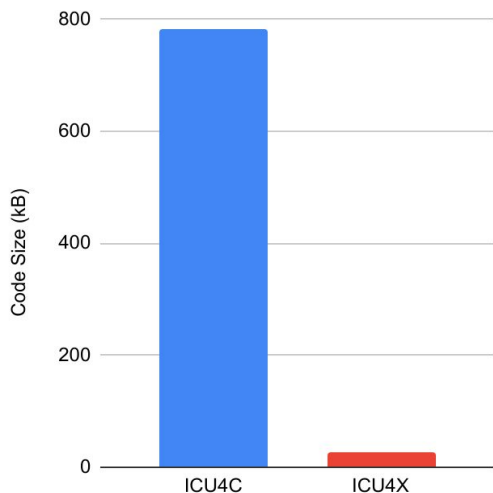
Enter a number

3.141

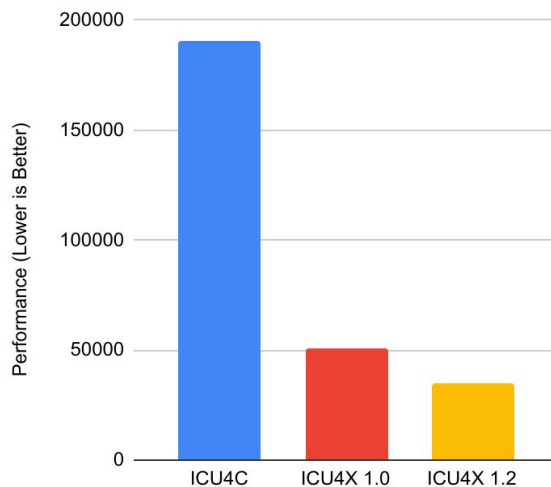
Formatted

ICU4X is Tiny and Fast

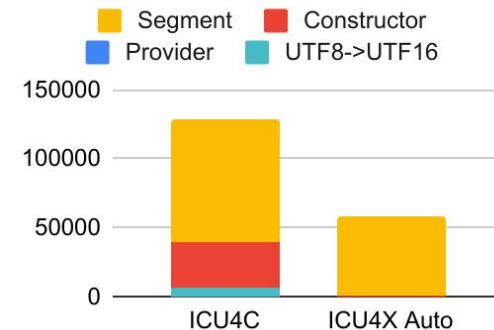
Basic Number Format: Code Size



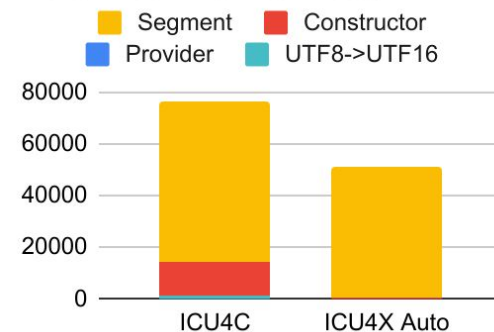
Basic Number Format: Performance



Word Break Perf: Non-Complex



Word Break Perf: Chinese



Building Your First App Using ICU4X

How This Tutorial Will Work

- Get your laptops out!
- The tutorial can be done in **Rust** or **JavaScript**

What We Will Build

In **Rust**:

- A command-line app that takes a locale and style and formats today's date.

In **JavaScript**:

- A web UI that takes a locale and style and formats a date from a date picker

APIs we will use

In **Rust**:

- `icu::locid::Locale`, `icu::datetime::DateFormatter`
- Explore the docs at <http://docs.rs/icu>

In **JavaScript**:

- `ICU4XLocale`, `ICU4XDateFormatter`
- Explore the docs at <https://unicode-org.github.io/icu4x/docs/ffi/js>

<https://tinyurl.com/2fnszkm9>
(20 minutes)

Recap

1. ICU4X is easily deployed in Rust, JavaScript and beyond
2. We implement CLDR standards for date formatting
3. ICU4X is well documented with many examples

Solutions:

Part 3

<https://jsfiddle.net/ye3k09c8/>

<https://gist.github.com/rust-play/5ad0bed1c089f5e202cc05b85ba9bf71>

Part 4

<https://jsfiddle.net/ye3k09c8/1/>

<https://gist.github.com/rust-play/2e5c190df50cd70f1ea44e8fa4c3f572>

Managing data in ICU4X

Data provider

```
trait DataProvider<M: KeyedDataMarker> {  
    fn load(&self, DataRequest) -> Result<M::Value, DataError>;  
}
```

1. The data provider trait (~interface) underlies everything ICU4X does with data
2. `KeyedDataMarker` represents an atomic unit of locale-specific data, associated with a data key
 - a. For example "Gregorian date symbols"
3. The trait abstracts the actual data loading mechanism away

Data provider

— — —

```
impl DateFormatter {  
    pub fn try_new_with_unstable<P>(&P, &DataLocale) -> Result<Self, Error>  
        where P: DataProvider<TimeSymbolsV1Marker>  
            + DataProvider<TimeLengthsV1Marker>  
            + DataProvider<OrdinalV1Marker>  
            + ...  
    { ... }  
}
```


Concrete data providers

1. You can implement your own data providers if you want to load data from custom sources
2. ICU4X comes with multiple implementations for different scenarios:
 - a. *Baked data* writes the data directly into Rust code. This is zero-overhead, and can be compiler optimized.
 - b. *Blob data* uses a platform independent binary serialization scheme. This allows loading data at runtime.
 - c. *Provider adapters* are available to combine multiple adapters for powerful runtime data pipelines.


Baked data

Struct `icu::datetime::provider::Baked`

[source](#) · [\[-\]](#)

```
pub struct Baked;
```

[\[-\]](#) Baked data

 This code is considered unstable; it may change at any time, in breaking or non-breaking ways, including in SemVer minor releases. In particular, the `DataProvider` implementations are only guaranteed to match with this version's `*_unstable` providers. Use with caution.

Trait Implementations

- [\[+\]](#) `impl DataProvider<BuddhistDateLengthsV1Marker> for Baked` [source](#)
- [\[+\]](#) `impl DataProvider<BuddhistDateSymbolsV1Marker> for Baked` [source](#)
- [\[+\]](#) `impl DataProvider<ChineseDateLengthsV1Marker> for Baked` [source](#)
- [\[+\]](#) `impl DataProvider<ChineseDateSymbolsV1Marker> for Baked` [source](#)
- [\[+\]](#) `impl DataProvider<CopticDateLengthsV1Marker> for Baked` [source](#)
- [\[+\]](#) `impl DataProvider<CopticDateSymbolsV1Marker> for Baked` [source](#)
- [\[+\]](#) `impl DataProvider<DangiDateLengthsV1Marker> for Baked` [source](#)
- [\[+\]](#) `impl DataProvider<DangiDateSymbolsV1Marker> for Baked` [source](#)
- [\[+\]](#) `impl DataProvider<DateSkeletonPatternsV1Marker> for Baked` [source](#)
- [\[+\]](#) `impl DataProvider<EthiopianDateLengthsV1Marker> for Baked` [source](#)
- [\[+\]](#) `impl DataProvider<EthiopianDateSymbolsV1Marker> for Baked` [source](#)
- [\[+\]](#) `impl DataProvider<ExemplarCitiesV1Marker> for Baked` [source](#)

```
DateFormatter::try_new_unstable(  
    icu::datetime::provider::Baked,  
    locale,  
)
```

Blob data

```
DateFormatter::try_new_with_buffer_provider(  
    BlobDataProvider::try_new_from_blob(  
        [0x00, 0x04, 0x04, 0x81, ...].into(),  
    ).unwrap(),  
    locale,  
)
```

```
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Decoded Text  
00000000 00 04 04 81 F5 5C 04 1B 00 00 00 E3 01 1B 00 00 . . . . \ . . . . .  
00000010 00 00 00 00 00 02 00 00 00 04 00 00 00 06 00 00 . . . . .  
00000020 00 0B 00 00 00 0D 00 00 00 0F 00 00 00 15 00 00 . . . . .  
00000030 00 1B 00 00 00 21 00 00 00 27 00 00 00 2C 00 00 . . . . ! . . . . .  
00000040 00 3C 00 00 00 48 00 00 00 4A 00 00 00 4C 00 00 < . . . H . . . J . . . L . . .  
00000050 00 4E 00 00 00 50 00 00 00 52 00 00 00 5E 00 00 . N . . . P . . . R . . . ^ . . .  
00000060 00 60 00 00 00 62 00 00 00 64 00 00 00 66 00 00 . . . b . . . d . . . f . . .  
00000070 00 68 00 00 00 6F 00 00 00 71 00 00 00 62 6E 63 . h . . . o . . . q . . . . b n c  
00000080 73 64 65 64 65 2D 41 54 65 6C 65 6E 65 6E 2D 30 s d e d e - A T e l e n e n - 0  
00000090 30 31 65 6E 2D 30 30 32 65 6E 2D 30 31 39 65 6E 0 1 e n - 0 0 2 e n - 0 1 9 e n  
000000A0 2D 31 34 32 65 6E 2D 47 42 65 6E 2D 47 42 2D 75 - 1 4 2 e n - G B e n - G B - u  
000000B0 2D 73 64 2D 67 62 65 6E 67 65 6E 2D 78 2D 72 65 - s d - g b e n g e n - x - r e  
000000C0 76 65 72 73 65 65 6F 66 61 66 69 69 73 6A 61 6A v e r s e e o f a f a i i s j a j  
000000D0 61 2D 78 2D 72 65 76 65 72 73 65 6C 61 70 74 72 a - x - r e v e r s e l a p t r  
000000E0 6F 72 75 73 72 73 72 2D 4C 61 74 6E 76 69 7A 68 o r u s r s r - l a t n v i z h  
000000F0 1C 01 18 02 01 07 19 03 10 0D 0E 0F 11 1A 04 0C . . . . .  
00000100 13 08 0B 14 15 00 05 06 12 17 0A 16 09 CF 04 1B . . . . .  
00000110 00 00 00 00 00 00 00 0B 00 00 00 16 00 00 00 22 . . . . . " . . . . .  
00000120 00 00 02 E 00 00 00 3A 00 00 00 46 00 00 00 52 . . . . . : . . . F . . . R . . .  
00000130 00 00 05 E 00 00 00 6A 00 00 00 77 00 00 00 85 . . . . . ^ . . . j . . . w . . .  
00000140 00 00 09 00 00 00 A3 00 00 00 B3 00 00 00 C3 . . . . .  
00000150 00 00 0D 00 00 00 E6 00 00 00 FA 00 00 00 0F . . . . .  
00000160 01 00 00 24 01 00 00 3A 01 00 00 50 01 00 00 67 . . . $ . . . : . . . P . . . g . . .  
00000170 01 00 00 81 01 00 00 9B 01 00 00 B7 01 00 00 0A . . . . .  
00000180 41 76 65 2C 20 6D 75 6E 64 65 0A 48 61 6C 6C 6F A v e , m u n d e . H a l l o  
00000190 20 57 65 6C 74 0B 41 68 6F 6A 20 73 76 C4 9B 74 W e l t . A h o j s v . t t  
000001A0 65 0B 48 65 6C 6C 6F 20 57 6F 72 6C 64 0B 4F 6C e . H e l l o W o r l d . O l  
000001B0 6C 65 68 20 44 6C 72 6F 77 0B 4F 6C C3 A1 2C 20 l e h D l r o w . O l . . .  
000001C0 6D 75 6E 64 6F 0B 53 61 6C 75 74 2C 20 6C 75 6D m u n d o . S a l u t , l u m  
000001D0 65 0B 53 65 72 76 75 73 20 57 65 6C 74 0B 68 65 e . S e r v u s W e l t . h e  
000001E0 69 20 6D 61 61 69 6C 6D 61 0C E4 BD A0 E5 A5 BD i m a i l m a . . . . .  
000001F0 E4 B8 9E E7 95 8C 0D 50 6F 7A 64 72 61 76 20 73 . . . . . P o z d r a v s  
00000200 76 65 74 65 0E 48 61 6C 6C C3 B3 2C 20 68 65 69 v e t e . H a l l . . , h e i  
00000210 6D 75 72 0E 53 61 6C 75 74 6F 6E 2C 20 4D 6F 6E m u r . S a l u t o n , M o n  
00000220 64 6F 0F 48 65 6C 6C 6F 20 66 72 6F 6D 20 F0 9F d o . H e l l o f r o m . . .  
00000230 8C 8D 0F 48 65 6C 6C 6F 20 66 72 6F 6D 20 F0 9F . . . H e l l o f r o m . . .  
00000240 8C 8E 0F 48 65 6C 6C 6F 20 66 72 6F 6D 20 F0 9F . . . H e l l o f r o m . . .  
00000250 8C 8F 12 48 65 6C 6C 6F 20 66 72 6F 6D 20 F0 9F . . . H e l l o f r o m . . .  
00000260 97 BA EF B8 8F 13 48 65 6C 6C 6F 20 66 72 6F 6D . . . . . H e l l o f r o m . . .  
00000270 20 F0 9F 87 AC F0 9F 87 A7 14 D0 9F D1 80 D0 B8 . . . . .  
00000280 D0 B2 D0 B5 D1 82 2C 20 D0 BC D0 B8 D1 80 14 D8 . . . . . , . . . . .  
00000290 B3 D9 84 D8 A7 D9 85 20 D8 AF D9 86 DB 8C D8 A7 . . . . .
```

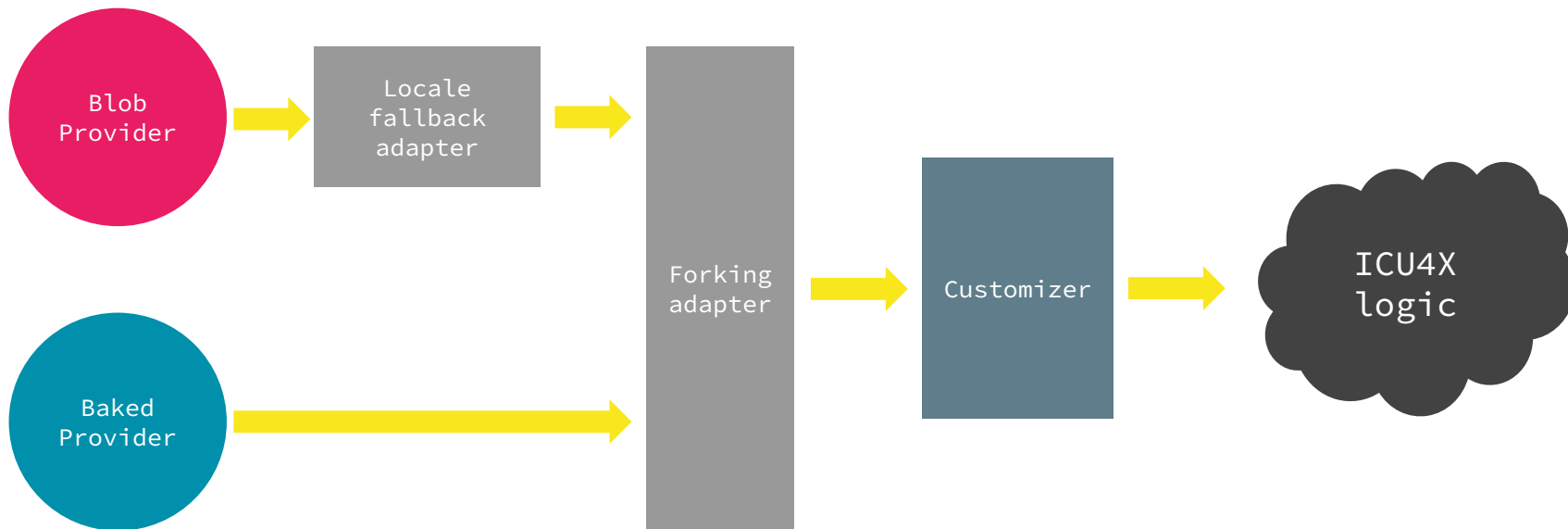
Blob data

— — —

```
trait DynamicDataProvider<M: DataMarker> {  
    fn load_data(&self, DataKey, DataRequest) -> Result<M::Value, Error>;  
}
```

```
trait BufferProvider {  
    fn load_buffer(&self, DataKey, DataRequest) -> Result<&[u8], Error>;  
}
```

Provider adapters



Generating data

1. If compiled data is too limiting, users can generate custom data
 - a. Compiled data includes hundreds of locales, which increases binary size
2. Data is sliced by locale (user selection) and data key (automatic)
3. Data can be tailored depending on runtime usage, e.g. with respect to fallback

<https://tinyurl.com/348kum7s>
(20 minutes)

Recap

1. We can dynamically load language packs
2. Use static analysis to generate sufficient data
3. Narrower APIs allow generating smaller data

Solutions:

<https://jsfiddle.net/ye3k09c8/2/>

<https://gist.github.com/rust-play/728adcbbaf022b7fc6f393c7cfbe83aa>

DateTimeFormatter and Data Size Tradeoff

1. **DateTimeFormatter**: formats best calendar for locale
 - a. "best" is either the CLDR default for that locale or the `-u-ca` extension
 - b. Fails if input is not either `DateTime<Iso>` or `DateTime<C>` where `C` is the calendar that matches the locale
 - c. Best for i18n correctness
2. **TypedDateTimeFormatter<C>**
 - a. Smaller data size given business requirements
 - b. Only accepts `DateTime<C>` where `C` is the same as the type parameter

Gregorian

other calendars

collation, segmentation, ...

Questions?