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 বুধ
- zh: 2023年三月8
- ja: R5/4/27

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

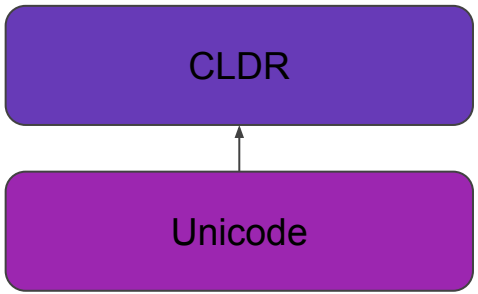
中|ワ|況|写|イノナ|開|億|が|ゃ|へ|者|43|俳|寺|式|7|沢|暮|ル|材|年|る|あん|移|酔|むぎ|え|す|写|情|主|逃|69|引|ぎ|う|ぱ|何|81|昇|フネイマ|本|因|ぱ|不|真|え|断|候|ら|。|吉|群|コクチ|賀|伝|ツエ|別|写|ユリキ|施|見|ら|力|1|取|目|れ|づ|ぱ|ぞ|然|野|テ|倍|展|ぴ|る|て|け|京|答|週|ネコ|入|整|料|勉|ら|げ|。

The i18n Stack

Unicode:
Foundational algorithms, specification, character set

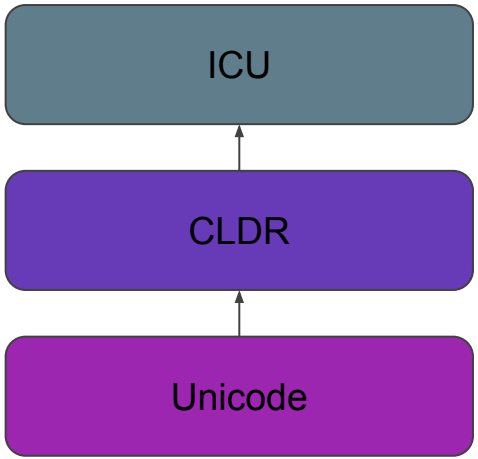
The i18n Stack

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



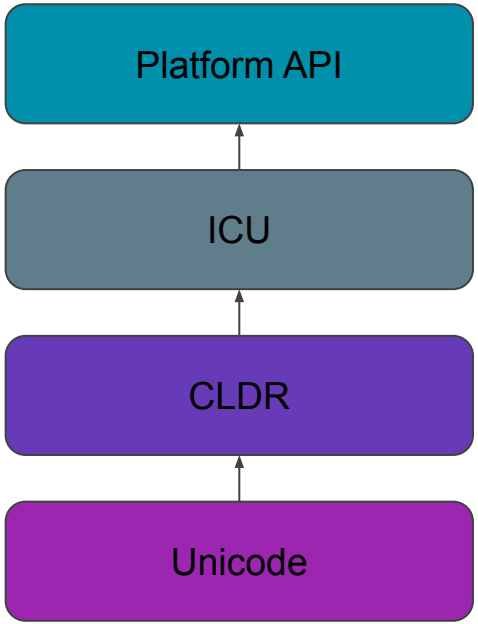
The i18n Stack

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

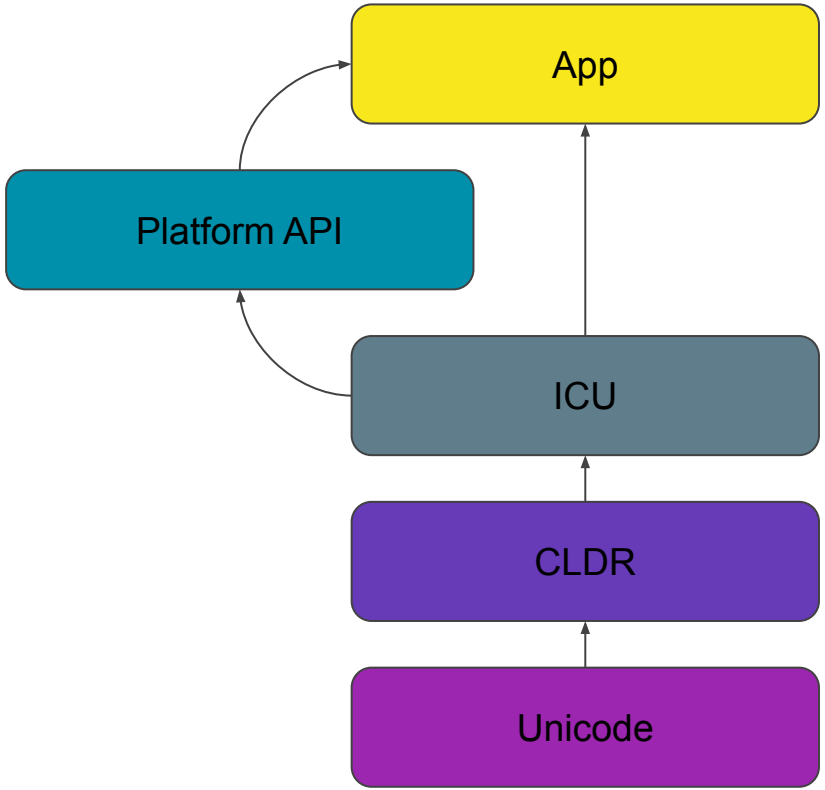


The i18n Stack

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

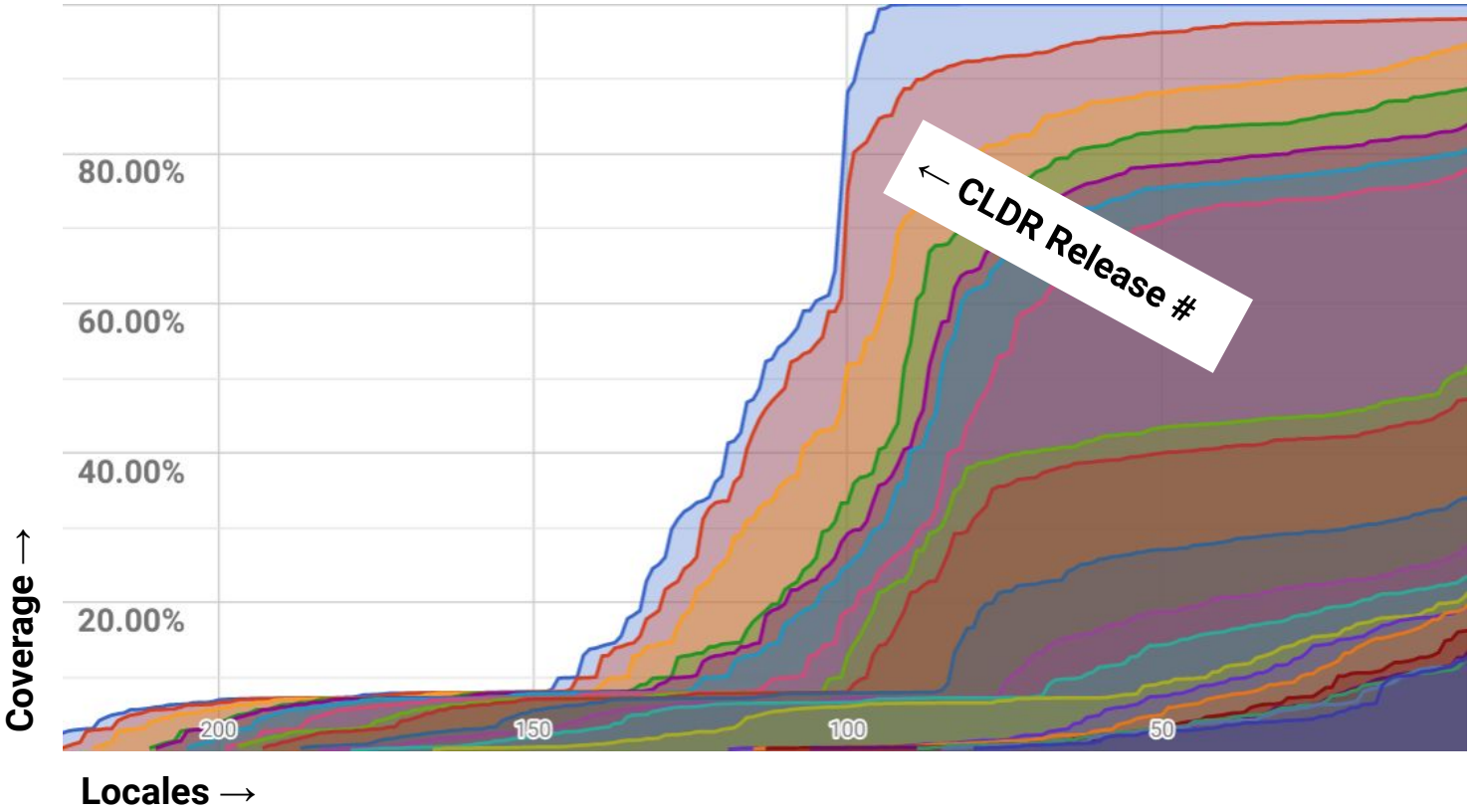


The i18n Stack

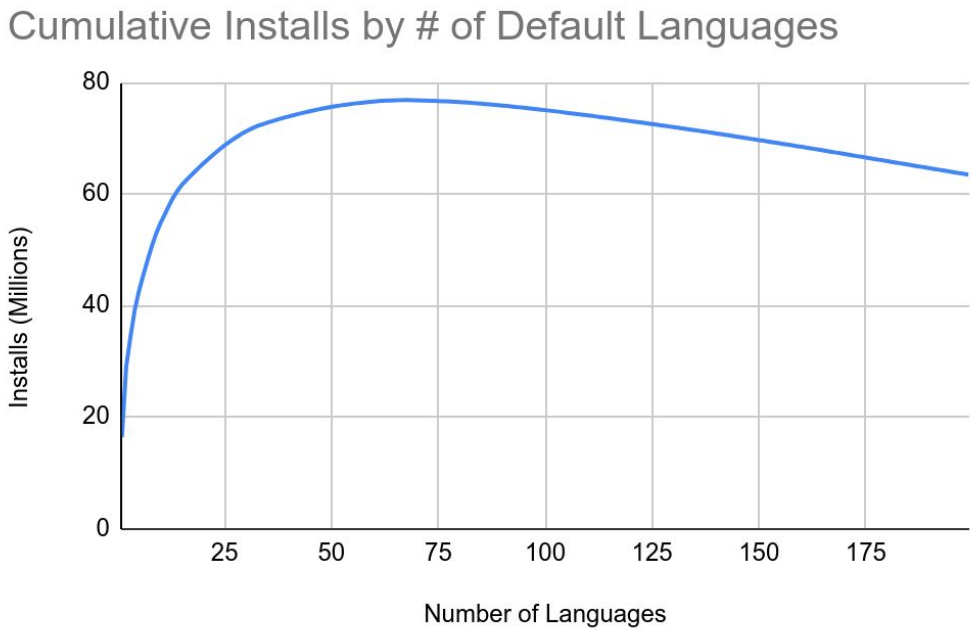


Challenges with Scaling i18n

Quadratic Growth of Data



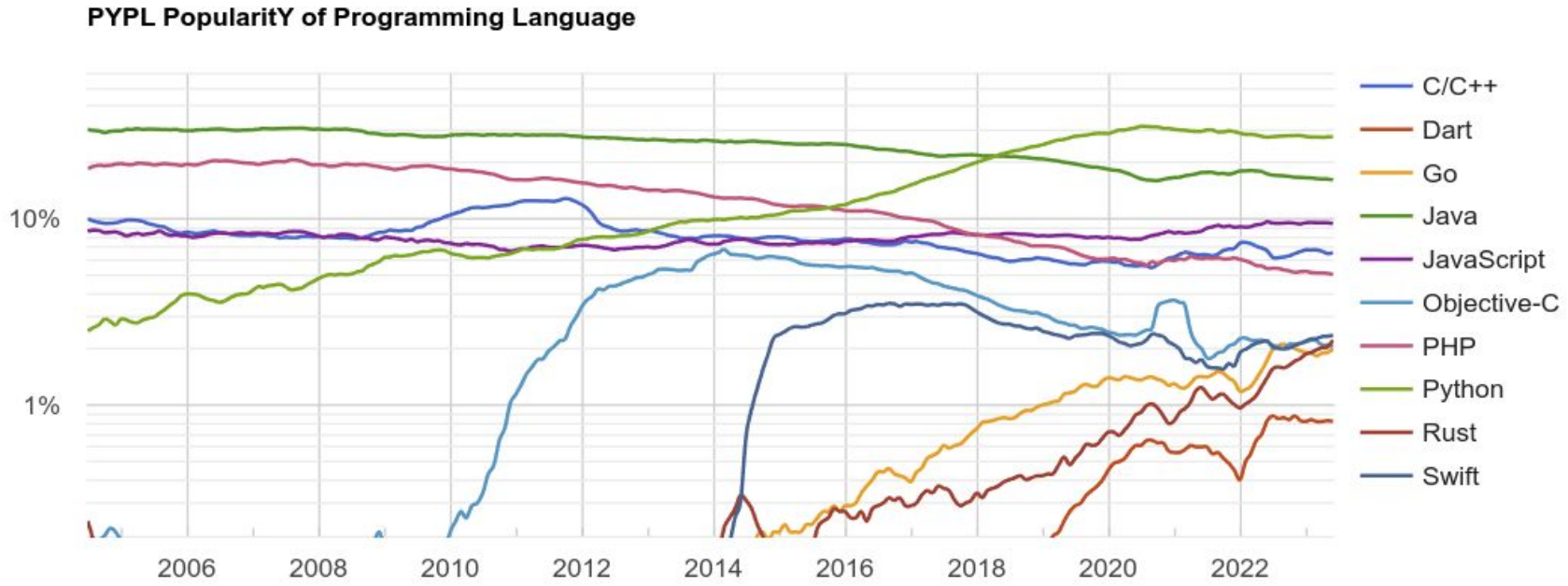
Adding more default languages has diminishing returns



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



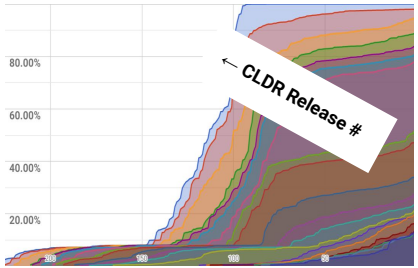
i18n-as-a-service doesn't make sense

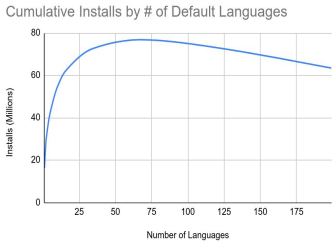
- Low latency requirements
- Data-heavy algorithms
- Privacy implications

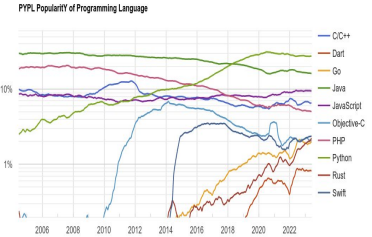
中|ワ|況|写|イノナ|開|億|が|ゃ|へ|者|43|俳|寺|式|7|沢|暮|ル|材|年|る|あん|移|酔|むぎ|え|す|写|情|主|逃|69|引|ぎ|う|ぱ|何|81|昇|フネイマ|本|因|ぱ|不|真|え|断|候|ら|。|吉|群|コクチ|賀|伝|ツエ|別|写|ユリキ|施|見|ら|力|1|取|目|れ|づ|ぱ|ぞ|然|野|テ|倍|展|ぴ|る|て|け|京|答|週|ネコ|入|整|料|勉|ら|げ|。

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|ゃ|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|ぱ|ぞ|然|野|テ|倍|展|び|る|て|け|京|答|週|ネコ|入|整|料|勉|ら|げ|。

Enter ICU4X

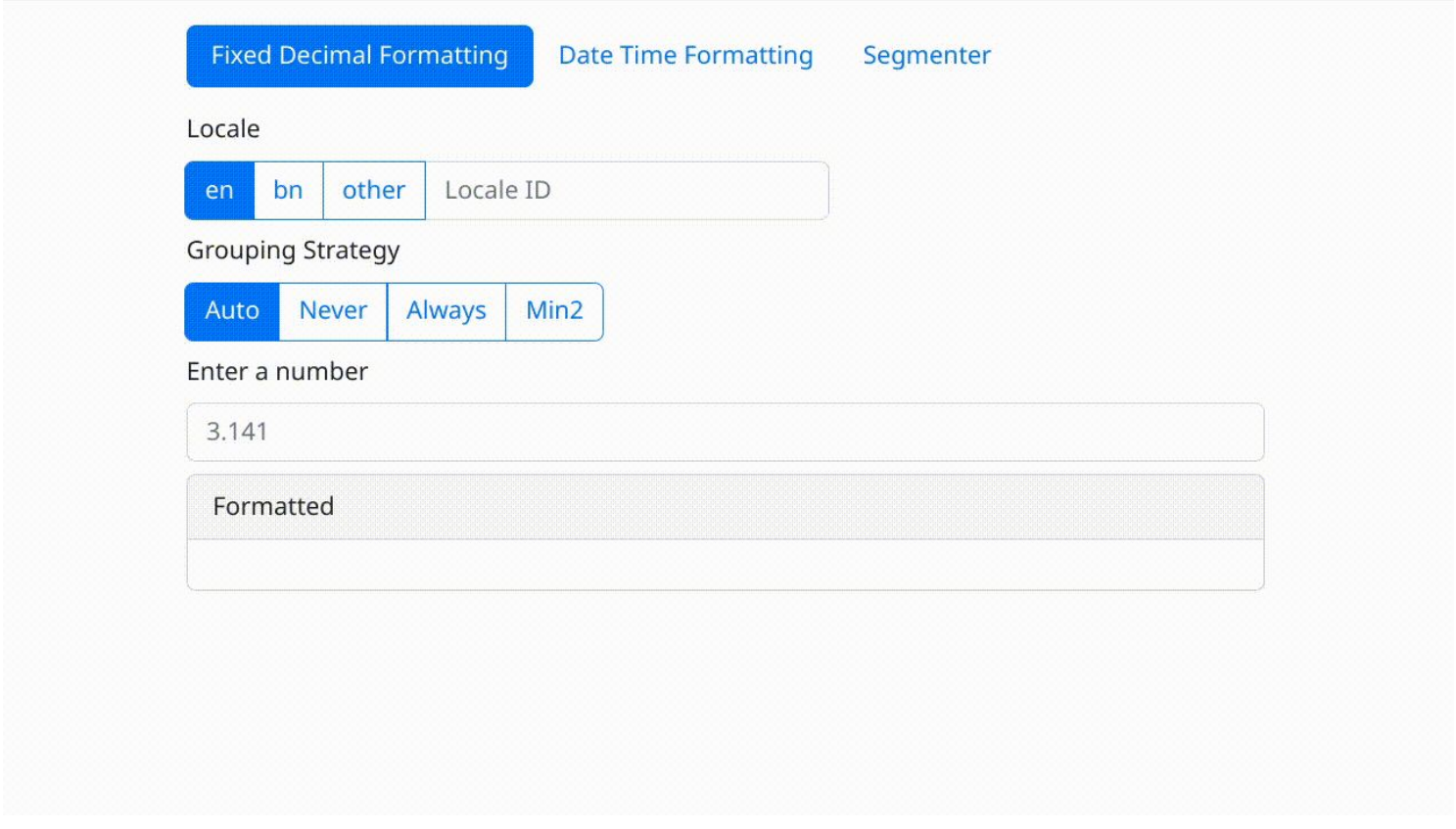
What is ICU4X?

ICU4X is an internationalization library that is:

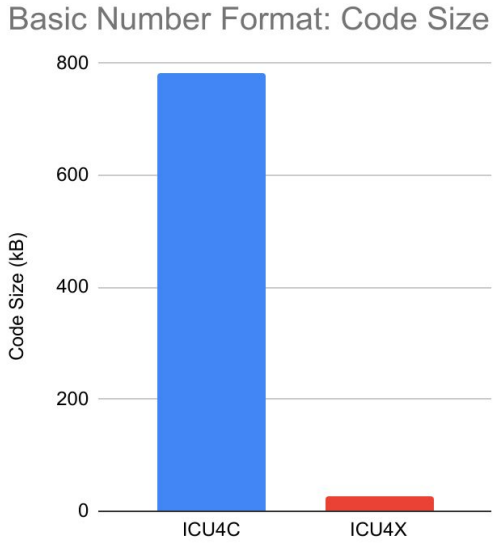
1. Portable by design
   - Runs on all types of devices with supported wrappers for multiple programming languages
2. Lightweight
   - Modular design, great for compile-time dead-code elimination and data slicing
3. Secure
   - Written in Rust, a memory-safe language

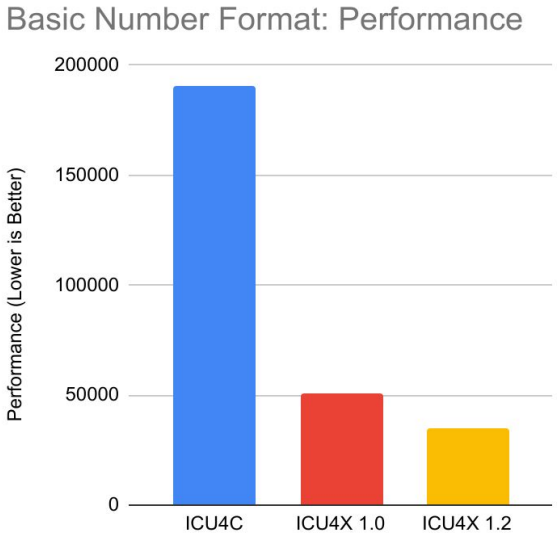
Who are we?

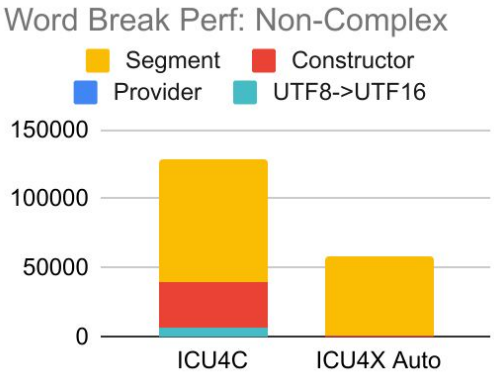
ICU4X WebAssembly Demo

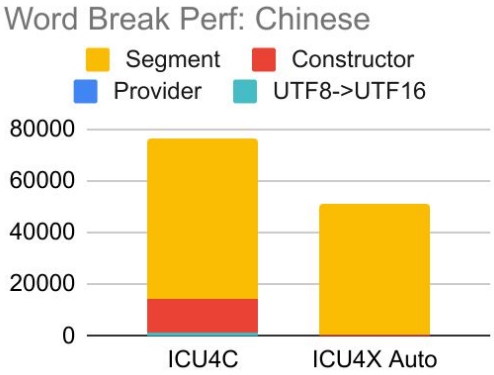


ICU4X is Tiny and Fast









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

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

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

Blob data

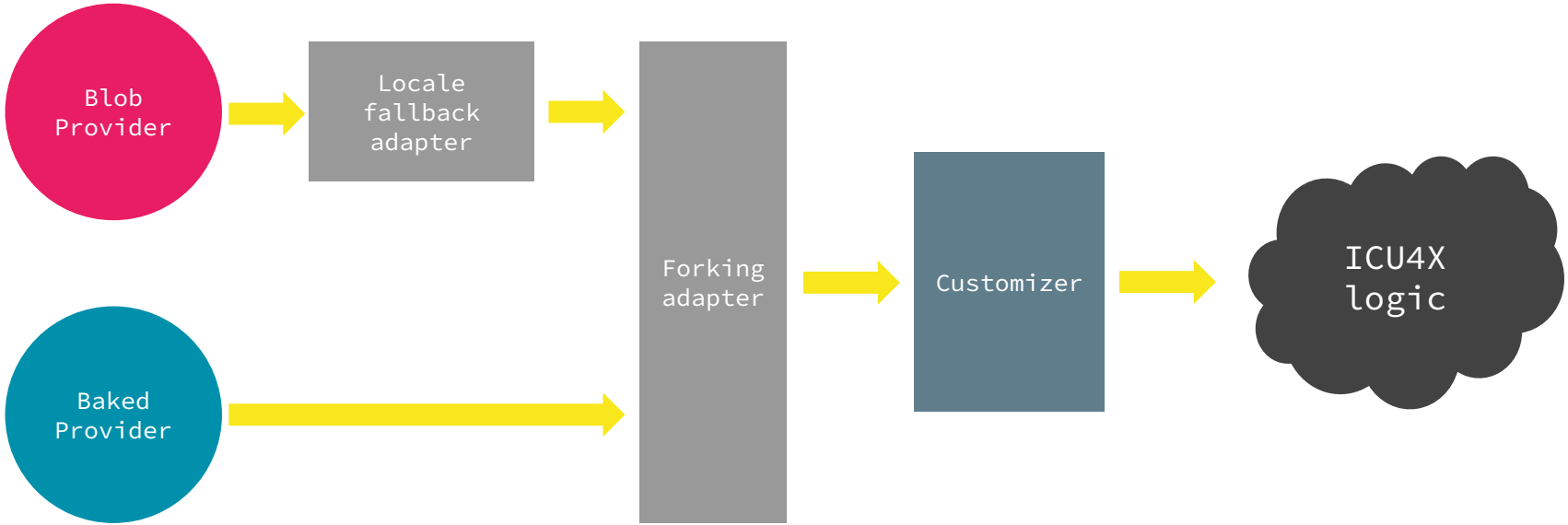
```
DateFormatter::try_new_with_buffer_provider(
  BlobDataProvider::try_new_from_blob(
   [0x00, 0x04, 0x04, 0x81, ...].into(),
  ).unwrap(),
  locale,
)
```

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

Questions?