

Modernizing Internationalization in Gecko and SpiderMonkey

Daniel Minor
SpiderMonkey / Mozilla

Overview

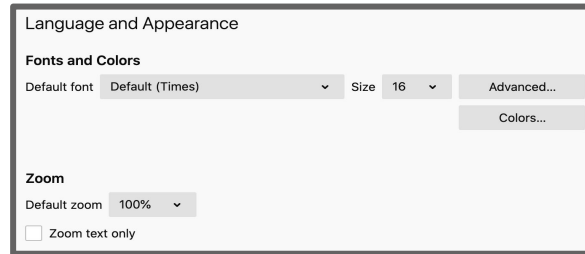
- What is internationalization?
- Deep Dive: Text Segmentation
- Localization in the browser
- How is all of this implemented?
- Experimenting with ICU4X in Firefox
- ICU4X Text Segmentation

What is Internationalization?

What is Internationalization?

- Part of a group of related ideas
 - Internationalization (i18n)
 - Translation
 - Localization (l10n)
- Let's talk about translation first

Translation



Translation

言語と外観

フォントと配色

既定のフォント 既定 (Times) ▼ サイズ 16 ▼ 詳細設定...

配色設定...

ズーム

既定のズーム 100% ▼

文字サイズのみ変更

भाषा और उपस्थिति

फ़ॉन्ट ब रंग

तयशुदा फ़ॉन्ट तयशुदा (Times) ▼ आकार 16 ▼ विस्तृत...

रंग...

रंग...

सूत्र

तयशुदा सूत्र 100% ▼

Zoom text only

اللغة و المظهر

الخطوط و الاوان

متقدم... ▼ 16 الحجم ▼ الخط المبني (Times) المبني المبني

الاولان...

التقريب

التقريب المبني %100 ▼

قرب النص فقط

Idioma y apariencia

Tipografías y colores

Tipografía predeterminada Predeterminada... ▼ Tamaño 16 ▼ Avanzadas...

Colores...

Ampliación

Ampliación predeterminada 100% ▼

Sólo ampliar texto

Language and Appearance

Fonts and Colors

Default font Default (Times) ▼ Size 16 ▼ Advanced...

Colors...

Zoom

Default zoom 100% ▼

Zoom text only

ಭಾಷೆ, ರೂಪರೇಖೆಗಳು

ಫಾಂಟ್ & ರಂಗಗಳು

ಅವಶ್ಯಕತೆಯ ಫಾಂಟ್ ಅವಶ್ಯಕತೆಯ (Times) ▼ ಪರಿಮಾಣ 16 ▼ ವಿಸ್ತೃತ...

ರಂಗ...

ರಂಗ...

ಜುಮ್

Default zoom 100% ▼

ಪಾಠ್ಯವನ್ನು ಮಾತ್ರವೇ ಹಾಕಿ ದಿವು

ენა და იერსახე

შრიფტები და ფერები

შრიფტის შერჩევა ნაგულისხმევი (Times) ▼ ზომა 16 ▼ დამატებით...

ფერები...

ზომა

ნაგულისხმევი ზომა 100% ▼

მხოლოდ ტექსტის ზომა

שפה ותצוגה

גופנים וצבעים

גופן ברירת מחדל ברירת מחדל (Times) גודל 16 ▼ מתקדם...

צבעים...

צבעים...

תצוגה

תצוגה ברירת מחדל ברירת מחדל 100% ▼

שינוי גודל טקסט בלבד

ภาษาและลักษณะที่ปรากฏ

แบบอักษรและสี

แบบอักษรเริ่มต้น คำเริ่มต้น (Times) ▼ ขนาด 16 ▼ ชัด...

สี...

ซูม

ซูมเริ่มต้น 100% ▼

ซูมข้อความเท่านั้น

Localization

- Translation + Cultural Adaptation
- Appropriate use of colours, symbols, images, etc.
- E.g. [Bug 615495 - Hide nurse when locale is 'ja'](#)
 - Around 12 years ago Mozilla support site redesign featured a cartoon cat nurse
 - In Japanese culture, this had unintended erotic connotations



Internationalization

- Internationalization consists of all the things that can be done by computer
 - E.g. Dates, times, currency formatting
- Internationalization enables localization
 - Less work for translators
- Data driven
 - CLDR - Common Locale Data Repository
 - XML and json
 - Unicode Technical Report #35 defines the schema and interpretation of this data

Internationalization Examples

Numbering systems	1, 2, 3, 4, ...	٤, ٣, ٢, ١, ٠
Number groupings	133,000.00	1,33,000.00
Date & Time	June 5, 2022 at 4:46:55 PM	2022年6月5日 16:46:41
Calendar systems	Tuesday, January 12, 2021	28 Tevet 5781
Text segmentation	Lorem ipsum dolor sit amet, consectetur adipiscing elit.	ญเพราะการจะเป็นอิสระได้จะต้องมีกำลังที่มากกว่าแข็งแกร่งกว่า
Directionality	Liked by John and two more people.	Liked by الويب and two more people.
Lists	Mary, Nick and Liam	Mary, Nick ⵎⵏⵏⵓⵎ Liam
Plural Rules	one, other	zero, one, two, few, many, other
Currencies	\$123.45	123.45 \$US, \$١٢٣,٤٥

ECMA-402: Internationalization in JavaScript

- Set of APIs that implement internationalization in JavaScript
- Create a formatter by specifying a locale and some options
- And the formatter provides a `format()` method to do the work

```
const number = 123456.789;

new Intl.NumberFormat('de-DE', { style: 'currency', currency: 'EUR' }).format(number);
// expected output: "123.456,79 €"

// the Japanese yen doesn't use a minor unit
new Intl.NumberFormat('ja-JP', { style: 'currency', currency: 'JPY' }).format(number);
// expected output: " ¥123,457"

// limit to three significant digits
new Intl.NumberFormat('en-IN', { maximumSignificantDigits: 3 }).format(number);
// expected output: "1,23,000"
```

Why Localization and Internationalization?

- For Mozilla, it's part of our manifesto, "The internet is a global public resource that must remain open and accessible"
- An English only web is neither open nor accessible
 - nor Chinese, Spanish, Arabic, etc.

Why Localization and Internationalization?




- Ensure that people are able to access the web in their own languages
 - Localize the browser
- Provide people with the tools they need to localize the web
 - ECMA-402 support in JavaScript

Deep Dive: Text Segmentation

Text Segmentation

- The process of chunking text into meaningful units
 - On character boundaries
 - On word boundaries
 - E.g. jump to the next word or sentence in a text editor
 - On line boundaries
 - E.g. for word wrapping text in a column

Grapheme Breaking: Code Points vs. Graphemes

- Graphemes are what are rendered to the screen
 - 
- Code points encode characters
- Not necessarily a 1:1 mapping from code points to graphemes
 -  U+1F44D THUMBS UP SIGN
 -  U+1F3FE EMOJI MODIFIER FITZPATRICK TYPE-5

Segmentation - Grapheme breaking

```
var string = "thumbs 👍 up"  
// 'thumbs 👍 up'  
  
[...s.segment(string)].map(s => s.segment)  
// (11) ['t', 'h', 'u', 'm', 'b', 's', ' ', '👍', ' ', 'u', 'p']  
  
[...string]  
// (12) ['t', 'h', 'u', 'm', 'b', 's', ' ', '👍', '▣', ' ', 'u', 'p']
```


Segmentation - Word breaking

- Break on word boundaries
- Complexity depends upon the language
 - A lot easier when there's spaces between words
 - More complicated for Asian languages

Word Breaking - Spanish

Es tan corto el amor, y es tan largo el olvido.

Word Breaking - Spanish

|Es|tan|corto|el|amor,|y|es|tan|largo|el|olvido.|

Word Breaking - Japanese

古池や 蛙飛び込む 水の音

Word Breaking - Japanese

|古池|や| |蛙|飛|び|込|む| |水|の|音|

Intl.Segmenter

- Stage 4 Proposal for ECMA-402
 - Already implemented in Chromium and Safari
- Use case: implementing text editors in JavaScript, etc.
- Current proposal includes grapheme, word and sentence breaking
- But not line breaking:
 - Line breaking is part of a v2 of the proposal
 - To do it properly, need information on text size and position, not just break locations

Intl.Segmenter Example

```
> // Create a locale-specific word segmenter  
let segmenter = new Intl.Segmenter("ja", {granularity: "word"});
```

Intl.Segmenter Example

```
> // Create a locale-specific word segmenter
let segmenter = new Intl.Segmenter("ja", {granularity: "word"});

// Use it to get an iterator for a string
let input = "古池や 蛙飛び込む 水の音";
let segments = segmenter.segment(input);
```


Intl.Segmenter Example

```
> // Create a locale-specific word segmenter
let segmenter = new Intl.Segmenter("ja", {granularity: "word"});

// Use it to get an iterator for a string
let input = "古池や 蛙飛び込む 水の音";
let segments = segmenter.segment(input);

// Use that for segmentation!
for (let {segment, index, isWordLike} of segments) {
  console.log("segment at code units [%d, %d): «%s»%s",
    index, index + segment.length,
    segment,
    isWordLike ? " (word-like)" : ""
  );
}
```

Intl.Segmenter Example

```
> // Create a locale-specific word segmenter
let segmenter = new Intl.Segmenter("ja", {granularity: "word"});

// Use it to get an iterator for a string
let input = "古池や 蛙飛び込む 水の音";
let segments = segmenter.segment(input);

// Use that for segmentation!
for (let {segment, index, isWordLike} of segments) {
  console.log("segment at code units [%d, %d): «%s»%s",
    index, index + segment.length,
    segment,
    isWordLike ? " (word-like)" : ""
  );
}
```

segment at code units [0, 2): «古池» (word-like) [VM35:10](#)

segment at code units [2, 3): «や» (word-like) [VM35:10](#)

segment at code units [3, 4): « » [VM35:10](#)

segment at code units [4, 5): «蛙» (word-like) [VM35:10](#)

segment at code units [5, 9): «飛び込む» (word-like) [VM35:10](#)

segment at code units [9, 10): « » [VM35:10](#)

segment at code units [10, 11): «水» (word-like) [VM35:10](#)

segment at code units [11, 12): «の» (word-like) [VM35:10](#)

segment at code units [12, 13): «音» (word-like) [VM35:10](#)

< undefined

Localization in the Browser

Localization in the Browser

- ECMA-402 allows developers to use JavaScript to localize the web
- But the browser is also a product that needs to be localized!

We ship 133 Locales in Firefox!

Acehnese	Bosnian	Esperanto	Icelandic	Latgalian	Norwegian Bokmål	Scots	Tajik
Acholi	Breton	Estonian	Iloko	Latvian	Norwegian Nynorsk	Serbian	Tamil
Afrikaans	Bulgarian	Finnish	Indonesian	Ligurian	Occitan	Sicilian	Telugu
Albanian	Burmese	French	Interlingua	Lithuanian	Odia	Silesian	Thai
Arabic	Catalan	Frisian	Irish	Luganda	Pai pai	Sinhala	Tibetan
Aragonese	Catalan (Valencian)	Friulian	Italian	Luxembourgish	Persian	Slovak	Triqui
Armenian	Central Kurdish	Fulah	Ixil	Macedonian	Polish	Slovenian	Turkish
Armenian Classic	Chinese (China)	Gaelic, Scottish	Japanese	Maithili	Portuguese (Brazil)	Songhay	Ukrainian
Armenian Eastern	Chinese (Taiwan)	Galician	Kabyle	Malay	Portuguese (Portugal)	Sorbian, Lower	Urdu
Arpitan	Chinyanja	Georgian	Kannada	Malayalam	Punjabi	Sorbian, Upper	Uzbek
Assamese	Crimean Tatar	German	Kaqchikel	Manx	Purépecha	Spanish (Argentina)	Vietnamese
Asturian	Croatian	Greek	Kashmiri	Marathi	Quechua Chanka	Spanish (Chile)	Welsh
Azerbaijani	Czech	Guarani	Kazakh	Miahuatlán Zapotec	Romanian	Spanish (Mexico)	Wolof
Basque	Danish	Gujarati	Khmer	Mixteco Yucuhiti	Romansh	Spanish (Spain)	Xhosa
Belarusian	Dutch	Hebrew	Kichwa	Mixtepec Mixtec	Russian	Swahili	
Bengali	English (Canada)	Hindi	Korean	Náhuat Pipil	Santali (OI Chiki)	Swedish	
Bodo	English (Great Britain)	Hungarian	Lao	Nepali	Sardinian	Tagalog	

Project Fluent

- How we localize Firefox
- A localization system for natural-sounding translations
 - Developers can write in English
 - Translators can produce appropriate translations for more grammatically complicated languages :)

```
# Simple things are simple.
hello-user = Hello, {SuserName}!

# Complex things are possible.
shared-photos =
  {SuserName} {SphotoCount ->
    [one] added a new photo
  } to {SuserGender ->
    [male] his stream
    [female] her stream
  }
  *{other} their stream
  }.
```

SuserName

SuserGender male
 female
 unspecified

SphotoCount

hello-user Hello, Anne!

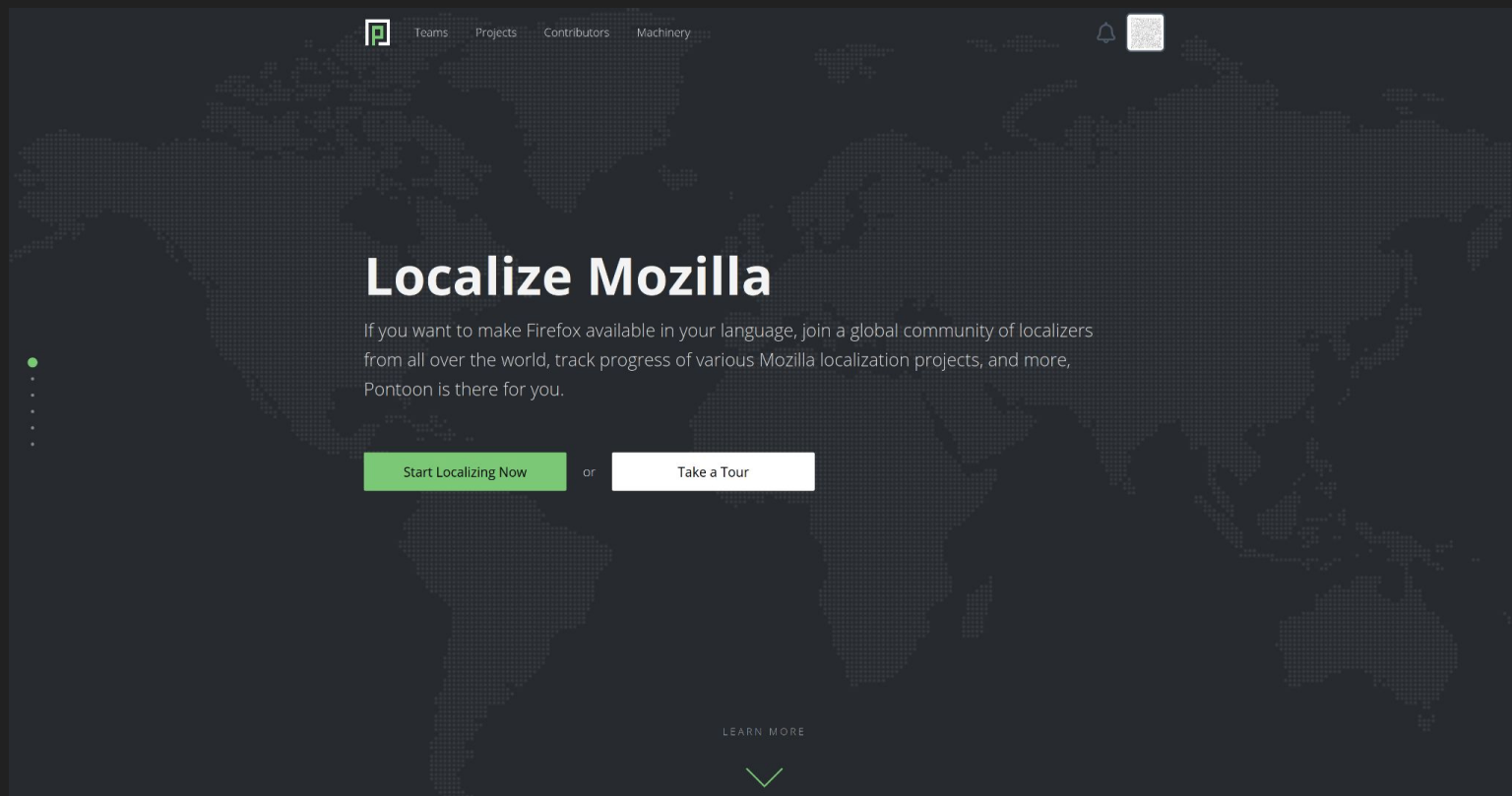
shared-photos Anne added 3 new photos to their stream.

Fluent for developers

- Developers write define the properties in English in text files
- Can be localized declaratively through the DOM using attributes
- Or programmatically through a JavaScript API

```
# Variables:
# $count (Number) - Number of tracking events blocked.
# $earliestDate (Number) - Unix timestamp in ms, representing a date. The
# earliest date recorded in the database.
graph-total-tracker-summary =
  { $count ->
    [one] <b>{ $count }</b> tracker blocked since { DATETIME($earliestDate, day: "numeric", month: "long", year: "numeric") }
    *[other] <b>{ $count }</b> trackers blocked since { DATETIME($earliestDate, day: "numeric", month: "long", year: "numeric") }
  }
```

Fluent for translators: Pontoon



Translating a string in Fluent

- UI Presents English source text
- Allows translator to edit target language text
 - Notice that there are more pluralizations in Lithuanian than in English

REQUEST CONTEXT or REPORT ISSUE

one ` { $count } ` tracker blocked since `{ DATETIME($earliestDate, day: "numeric", month: "long", year: "numeric") }`

other ` { $count } ` trackers blocked since `{ DATETIME($earliestDate, day: "numeric", month: "long", year: "numeric") }`

COMMENT Variables: \$count (Number) - Number of tracking events blocked. \$earliestDate (Number) - Unix timestamp in ms, representing a date. The earliest date recorded in the database.

CONTEXT: track total tracker summary

one (e.g. 1)	<code>Nuo { DATETIME(\$earliestDate, day: "numeric", month: "long", year: "numeric") } buvo užblokuotas</code>
few (e.g. 2)	<code>Nuo { DATETIME(\$earliestDate, day: "numeric", month: "long", year: "numeric") } buvo užblokuoti {</code>
other (e.g. 0)	<code>Nuo { DATETIME(\$earliestDate, day: "numeric", month: "long", year: "numeric") } buvo užblokuota {</code>

Internationalization in Fluent

- Fluent needs to localize numbers and dates too!
- Built-in functions NUMBER, DATETIME to handle this
- Based on ECMA-402 APIs

DATETIME

Formats a date to a string in a given locale.

Example:

```
today-is = Today is { DATETIME($date, month: "long", year: "numeric", day: "numeric") }
```

Parameters:

```
hour12  
weekday  
era  
year  
month  
day  
hour  
minute  
second  
timeZoneName
```

Developer parameters:

```
timeZone
```

See the [Intl.DateTimeFormat](#) for the description of the parameters.

How is this all Implemented?

ICU4C

- International Components for Unicode for C (ICU4C)
 - Also an ICU4J for Java
 - And an ICU4X that we'll discuss later
- Enormous C (and C++) i18n library, first released in 1999
- Used everywhere!
 - Implement internationalization in SpiderMonkey and Gecko
 - And in Chromium, WebKit, etc.
 - And in OS X, Linux, Adobe products, etc. etc.

ICU4C

- Provides a lot of functionality
- But it's monolithic
 - Very hard to remove code or data that you don't use
 - Large dependency for browsers
- API mismatch with ECMA-402
 - Lots of work setting up calls, handling options and converting results

Unifying Firefox's Internationalization Code

- At the beginning of 2021 Firefox effectively had three i18n implementations:
 - One in SpiderMonkey
 - One in Fluent
 - And one in Gecko
- ICU4C calls scattered throughout the code base
 - Lots of code duplication to handle parameters and return values
- No guarantee of consistent results between SpiderMonkey and Gecko
 - e.g. using different options when formatting numbers and dates in SpiderMonkey and in Fluent

Unifying Firefox's Internationalization Code

- Moved all ICU calls to a single library usable from SpiderMonkey and Gecko
- Developed interfaces based on ECMA-402
 - Hide ICU4C complexity from developers
 - Get rid of duplicated code
 - Make sure we're consistent when calling into ICU APIs
- Largely used existing SpiderMonkey code
 - SpiderMonkey code well tested, thanks to test262

Unification Results

- Got rid of a lot of duplicated code
- Had almost no regressions while doing so
- Full set of ECMA-402 APIs are now available to localize the browser
- And we can easily experiment with other internationalization libraries

Experimenting with ICU4X in Firefox

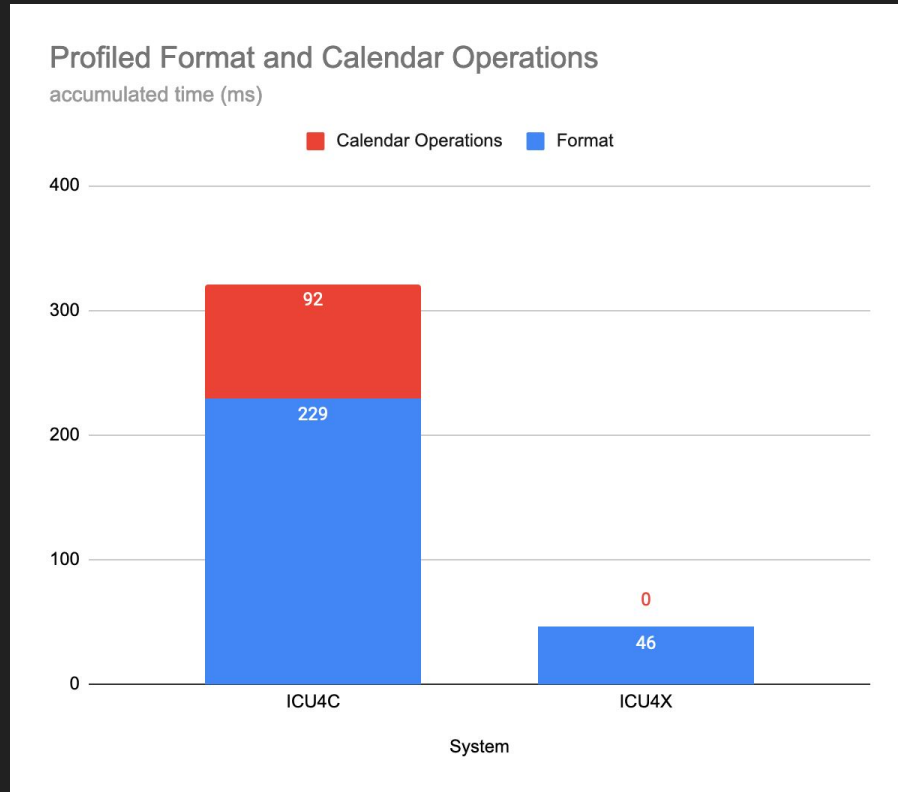
ICU4X Project

- Reimplementation of ICU4C designed around the needs of the web
 - Smaller, faster
 - Data is not monolithic
 - APIs based on ECMA-402
- Initial implementation in Rust, support for other languages through FFI
- Under development by Google and Mozilla since 2020
 - 1.0 Release planned for end of June 2022

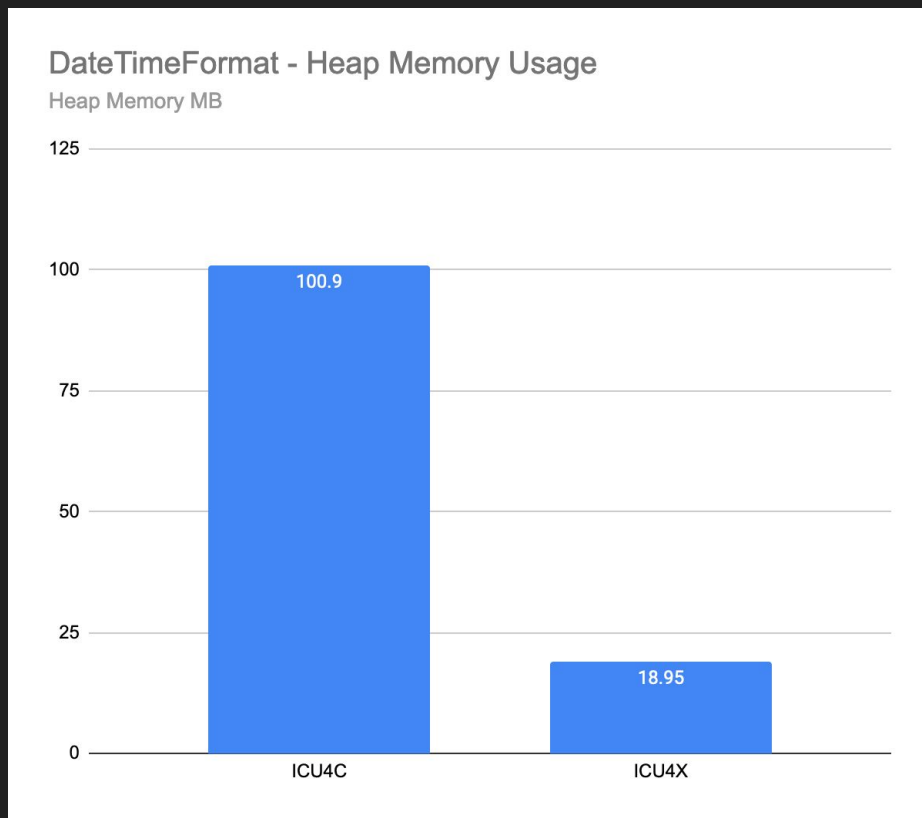
Experimenting with ICU4X in Firefox

- In June 2021, ran experiments using ICU4X in SpiderMonkey
 - Intl.NumberFormat
 - Intl.DateTimeFormat
 - Intl.PluralRules
 - Intl.Locale
- Looked at performance, memory use and correctness

Intl.DateTimeFormat Performance



Intl.DateTimeFormat Memory Use



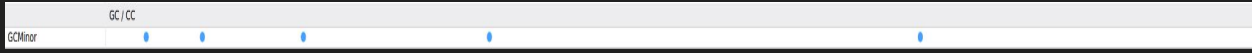
Correctness

- Ran correctness tests using Test-262 for:
 - Intl.Locale
 - Intl.PluralRules
- Other implementations were not complete enough for results to be meaningful
- Handful of failures
 - Mostly around locale canonicalization

Some Unexpected Results

- Intl.NumberFormat memory use was worse with ICU4X!
 - 78.0 MB with ICU4X vs. 21.3 MB with ICU4C
- Smaller ICU4X objects meant less frequent Garbage Collection

- ICU4X



- ICU4C:



- This was a side-effect of the integration with SpiderMonkey
 - Need to estimate object size for Unicode objects in SpiderMonkey
 - But this estimate affects the behaviour of the garbage collector
 - Not an ICU4X issue :)

Overall Results

- In general performance and memory use are much better
- But need to be cautious
 - Incomplete implementations of DateTimeFormat and NumberFormat
 - Integration was a little hacky
 - It would be interesting to benchmark again after ICU4X 1.0
- ICU4X was promising enough to continue our development
 - Collator
 - DateTimeFormat
 - Segmenter

ICU4X Text Segmentation

Using ICU4X for Segmentation in Firefox

- ICU4C is not suitable for Firefox segmentation
- Layout engine does not use ICU4C for segmentation in the browser
 - Need to adjust line breaking results according to CSS properties
 - ICU4C data size is too large, because each word-break and line-break combination requires its own data set.
- Using ICU4C for Intl.Segmenter but not layout would mean inconsistent results
 - Exactly what we wanted to avoid with our unification project

Using ICU4X for Segmentation in Firefox

- Segmentation is also a great use case for experimenting with ICU4X
 - Not just a faster version of ICU4C
 - Does something we can't currently do with ICU4C
- Will give us consistent segmentation:
 - Across platforms
 - Our Layout implementation is platform specific
 - And between Layout and SpiderMonkey

Implementing Segmentation in ICU4X

- Rule based
 - Algorithm determines whether a character is a break
 - UAX #14: Unicode Line Breaking Algorithm
 - UAX #29: Unicode Text Segmentation
- Dictionary based
 - Look up codepoint in a trie to determine where to break
 - Used for Asian languages
- Neural network models
 - LSTM models trained on dictionaries
 - Space/Time tradeoff: slower, but require less storage

Integrating the ICU4X Segmenter

- Will start after ICU4X 1.0 Release
- Already refactored existing Layout code to have an ECMA-402 like interface
- API surface is small
- But need to figure out how to handle data packaging in Firefox
 - Something we didn't try in our earlier experiments

Experimenting with the ICU4X Segmenter

- Write microbenchmarks for word and line breaking
- Analyze impact on code and data size
- Enable for Linux Nightly builds
 - Behind a preference
 - Linux segmentation is currently the least functional, so the lowest risk for us
 - Check for regressions, performance problems, etc.

Implementing Intl.Segmenter

- Start once we're happy with the ICU4X integration
- Should be straightforward
 - Have a work-in-progress implementation using ICU4C for an older version of the spec
- Validate against Test-262

Conclusions

- Internationalization is important for the browser and in JavaScript
- By unifying our implementation, we've simplified our code base significantly
- We also set ourselves up to try out new implementations
- Because of this work, we can easily try ICU4X for text segmentation

Questions?