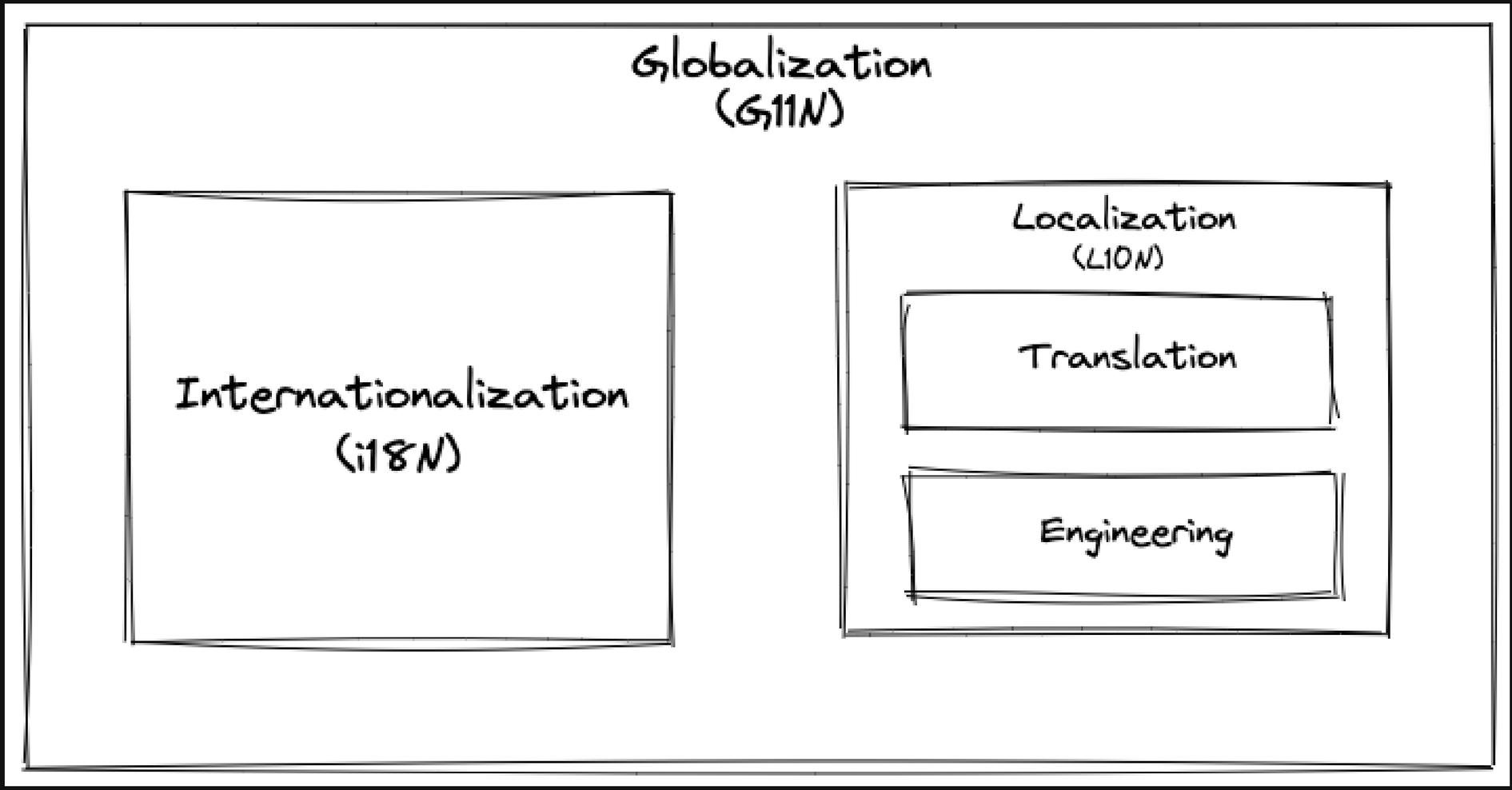
i18N in Java

Presented by @Dawei Ma

G11N/i18N/L10N



> Internationalization is the process of designing an application so that it can be adapted to various **languages** and **regions** without engineering changes.

An internationalized program has the following characteristics:

- With the addition of localized data, the same executable can run worldwide.
- Textual elements, such as status messages and the GUI component labels, are not hardcoded in the program. Instead they are stored outside the source code and retrieved dynamically.
- Support for new languages does not require recompilation.
- Culturally-dependent data, such as dates and currencies, appear in formats that conform to the end user's region and language.
- It can be localized quickly.

Java i18n Demo

```java
import java.util.Locale;
import java.util.ResourceBundle;

public class Hello {

    public static void main(String[] args) {
        String language = "en";
        String country = "US";

        if (args.length == 1) {
            language = args[0];
        } else if (args.length == 2) {
            language = args[0];
            country = args[1];
        }

        var locale = new Locale(language, country);
        var messages = ResourceBundle.getBundle("messages", locale);

        System.out.print(messages.getString("hello") + " ");
        System.out.println(messages.getString("world"));
    }
}
```

messages_en.properties

```
hello=Hello(en)
world=World
```

messages_en_US.properties

```
world=World(en_US)
```

messages_es.properties

```
hello=Hola
world=Mundo
```

execute java

```
java Hello.java
java Hello.java es
```

```
Hello(en) World(en_US)
Hola Mundo
```

Java i18N workflow

1. Create the Properties Files

```
greetings = Hello
farewell = Goodbye
inquiry = How are you?
```

2. Define the **Locale**

```
aLocale = new Locale("en","US");
```

3. Create a **ResourceBundle**

```
messages = ResourceBundle.getBundle("MessagesBundle", currentLocale);
```

```
MessagesBundle_en_US.properties
MessagesBundle_fr_FR.properties
MessagesBundle_de_DE.properties
```

4. Fetch the Text from the ResourceBundle

```
String msg1 = messages.getString("greetings");
```

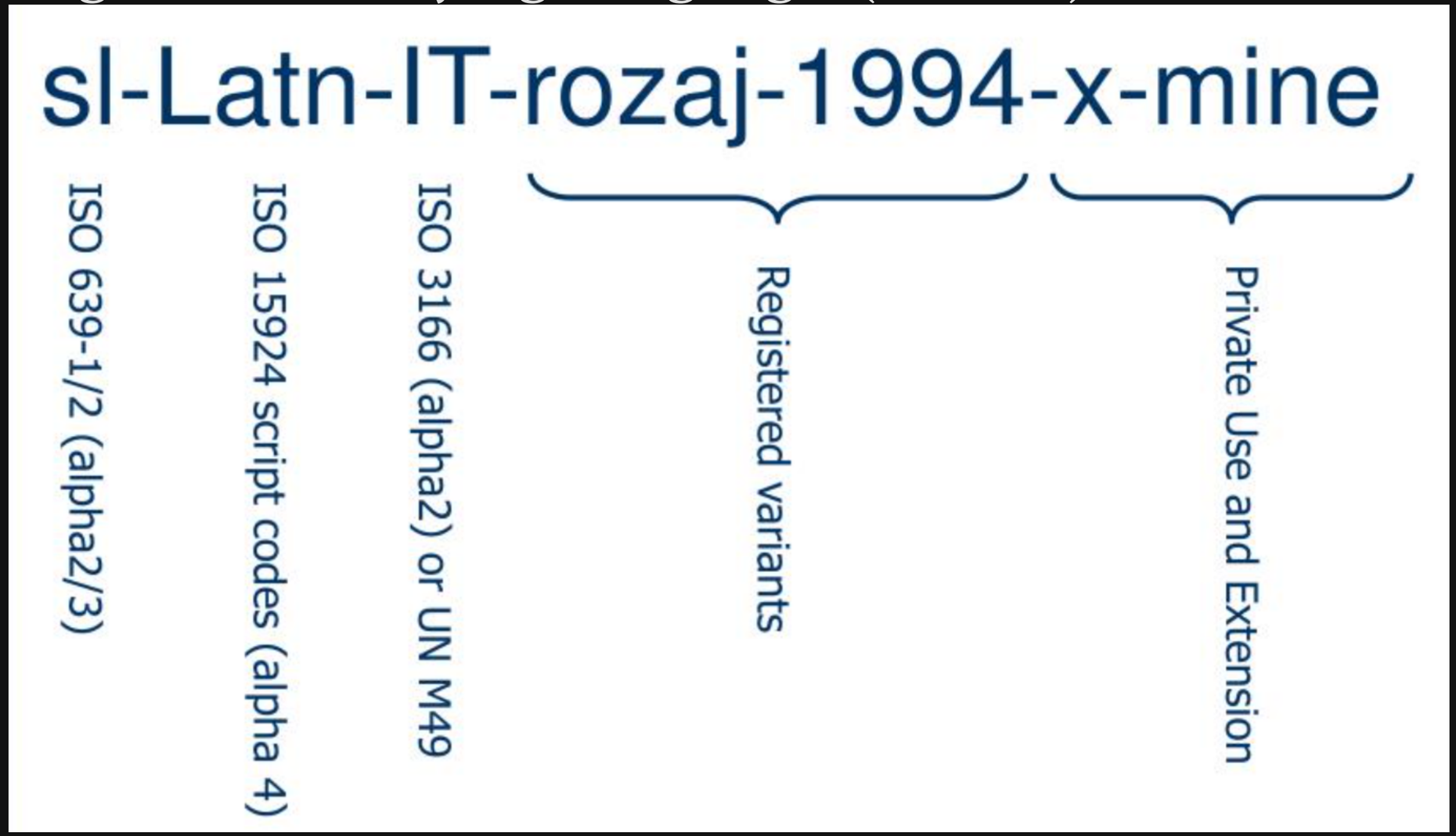
How does an internationalized program identify the appropriate language and region of its end users?

It references a Locale object.

Locale in i18N

> Locale is the user-specific location and cultural information managed by a computer. (RFC6365)
>
> A concept or identifier used by programmers to represent a particular collection of cultural, regional, or linguistic preferences.

Tags for Identifying Languages(BCP 47)




```
langtag = language["-" script]["-" region]*("-" variant)*("-" extension)["-" privateuse]
```

i18N IETF Standard

- Terminology Used in Internationalization in the IETF
- IETF BCP Best Current Practice 47 Language Tag Registry Update (LTRU)
- Making Sense of Language Tags

Locale in Java

java.util.Locale

- Implements IETF BCP 47
  - RFC 4647: Matching of Language Tags
    - *Filtering* is used to get all matching locales
    - *lookup* is to choose the best matching locale
  - RFC 5646: Tags for Identifying Languages
- Refers to IETF RFC 2616
  - Quality Values
- Refers to ISO
  - ISO 639: Language codes
  - ISO 3166: Country codes
  - ISO 15924: Script code

How does Java get messages by locale identify?

It’s *ResourceBundle*!

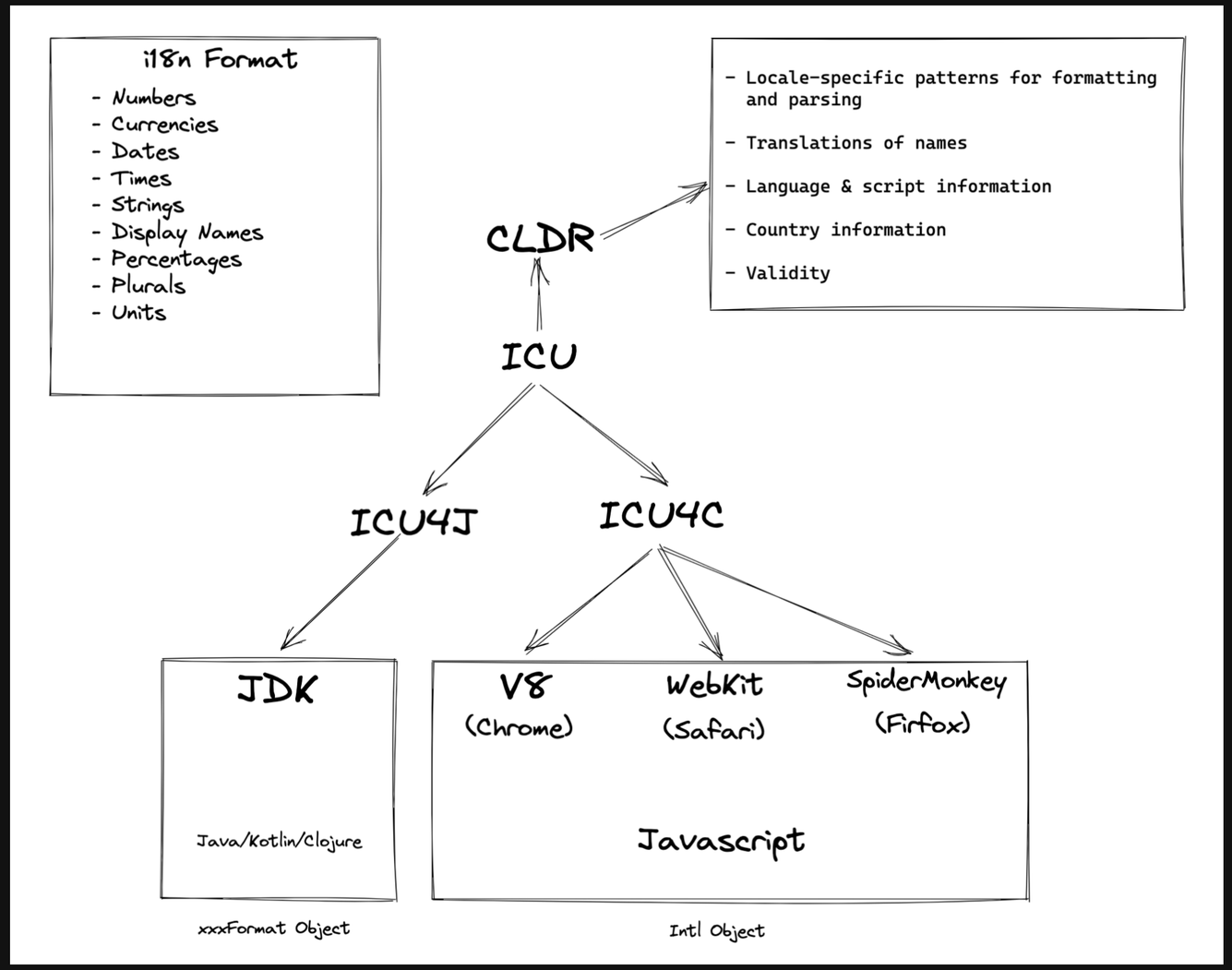
ResourceBundle in Java

java.util.ResourceBundle

Others for i18N

- Common Language Data Repository(CLDR)[Incorporated into JDK 8]
  - Locale-specific patterns for formatting and parsing: dates, times, timezones, numbers and currency values, measurement units,…
  - Translations of names: languages, scripts, countries and regions, currencies, eras, months, weekdays, day periods, time zones, cities, and time units, and sequences (and search keywords),…
  - Language & script information: characters used; plural cases; gender of lists; capitalization; rules for sorting & searching; writing direction; transliteration rules;…
  - Country information: language usage, currency information, calendar preference, week conventions,…
  - Validity: Definitions, aliases, and validity information for Unicode locales, languages, scripts, regions, and extensions,…
- UNICODE LOCALE DATA MARKUP LANGUAGE (LDML)

i18n Format


i18n Format
- Numbers
- Currencies
- Dates
- Times
- Strings
- Display Names
- Percentages
- Plurals
- Units
- Locale-specific patterns for formatting and parsing
- Translations of names
- Language & script information
- Country information
- Validity
CLDR
ICU
ICU4J
ICU4C
JDK
Java/Kotlin/Clojure
V8
(Chrome)
WebKit
(Safari)
SpiderMonkey
(Firfox)
Javascript
xxxFormat Object
Intl Object

i18n Format Reference

1. https://docs.oracle.com/javase/tutorial/i18n/format/index.html
2. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Intl
3. http://rxaviers.github.io/javascript-globalization/
4. https://lokalise.com/blog/comparing-libraries-translating-js-apps/
5. https://github.com/unicode-org/cldr-json#package-organization

Outside of Java

- GUN gettext
  - C
  - C++
  - Python
  - PHP
  - Elixir
- ICU
  - Code Page Conversion
  - Collation
  - Formatting(CLDR)
  - Time Calculations
  - Unicode Support
  - Regular Expression
  - Bidi

Further reading list

- Building a minimal i18n library
- Clojure uses standard Java ResourceBundle
- awesome-i18n 1
- awesome-i18n 2
- A Beginner’s Guide to Internationalization in Java
-
-
- ECMAScript® 2021 Internationalization API Specification
- iLib-js

Thanks!