Road to ICU:
Challenges and Best Practices of ICU Adoption from JDK

Pu Chen - Globalization Principal Engineer
Bo Yang - Globalization Development Manager
Teresa Marshall - VP, Globalization & Localization

Forward-Looking Statement

Statement under the Private Securities Litigation Reform Act of 1995

This presentation may contain forward-looking statements that involve risks, uncertainties, and assumptions. If any such uncertainties materialize or if any of the assumptions proves incorrect, the results of salesforce.com, inc. could differ materially from the results expressed or implied by the forward-looking statements we make. All statements other than statements of historical fact could be deemed forward-looking, including any projections of product or service availability, subscriber growth, earnings, revenues, or other financial items and any statements regarding strategies or plans of management for future operations, statements of belief, any statements concerning new, planned, or upgraded services or technology developments and customer contracts or use of our services.

The risks and uncertainties referred to above include – but are not limited to – risks associated with developing and delivering new functionality for our service, new products and services, our new business model, our past operating losses, possible fluctuations in our operating results and rate of growth, interruptions or delays in our Web hosting, breach of our security measures, the outcome of any litigation, risks associated with completed and any possible mergers and acquisitions, the immature market in which we operate, our relatively limited operating history, our ability to expand, retain, and motivate our employees and manage our growth, new releases of our service and successful customer deployment, our limited history reselling non-salesforce.com products, and utilization and selling to larger enterprise customers. Further information on potential factors that could affect the financial results of salesforce.com, inc. is included in our annual report on Form 10-K for the most recent fiscal year and in our quarterly report on Form 10-Q for the most recent fiscal quarter. These documents and others containing important disclosures are available on the SEC Filings section of the Investor Information section of our Web site.

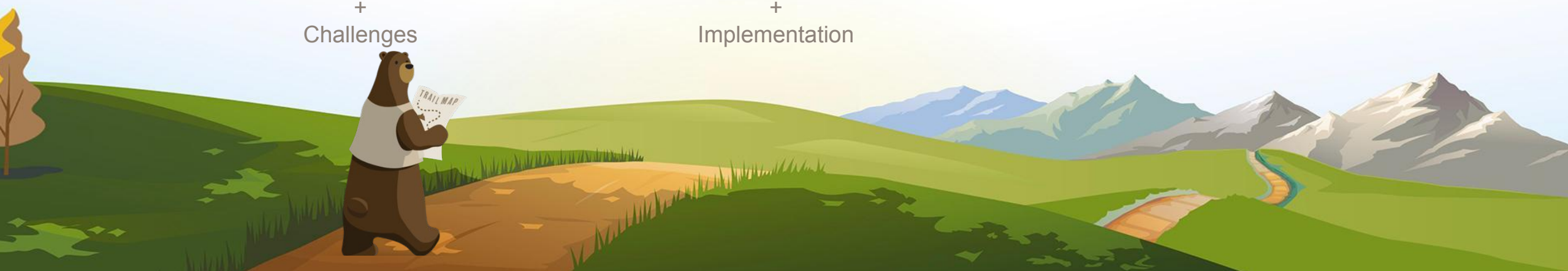
Any unreleased services or features referenced in this or other presentations, press releases or public statements are not currently available and may not be delivered on time or at all. Customers who purchase our services should make the purchase decisions based upon features that are currently available. Salesforce.com, inc. assumes no obligation and does not intend to update these forward-looking statements.

Path to ICU

Requirements
+
Challenges

Planning
+
Implementation

Case Studies

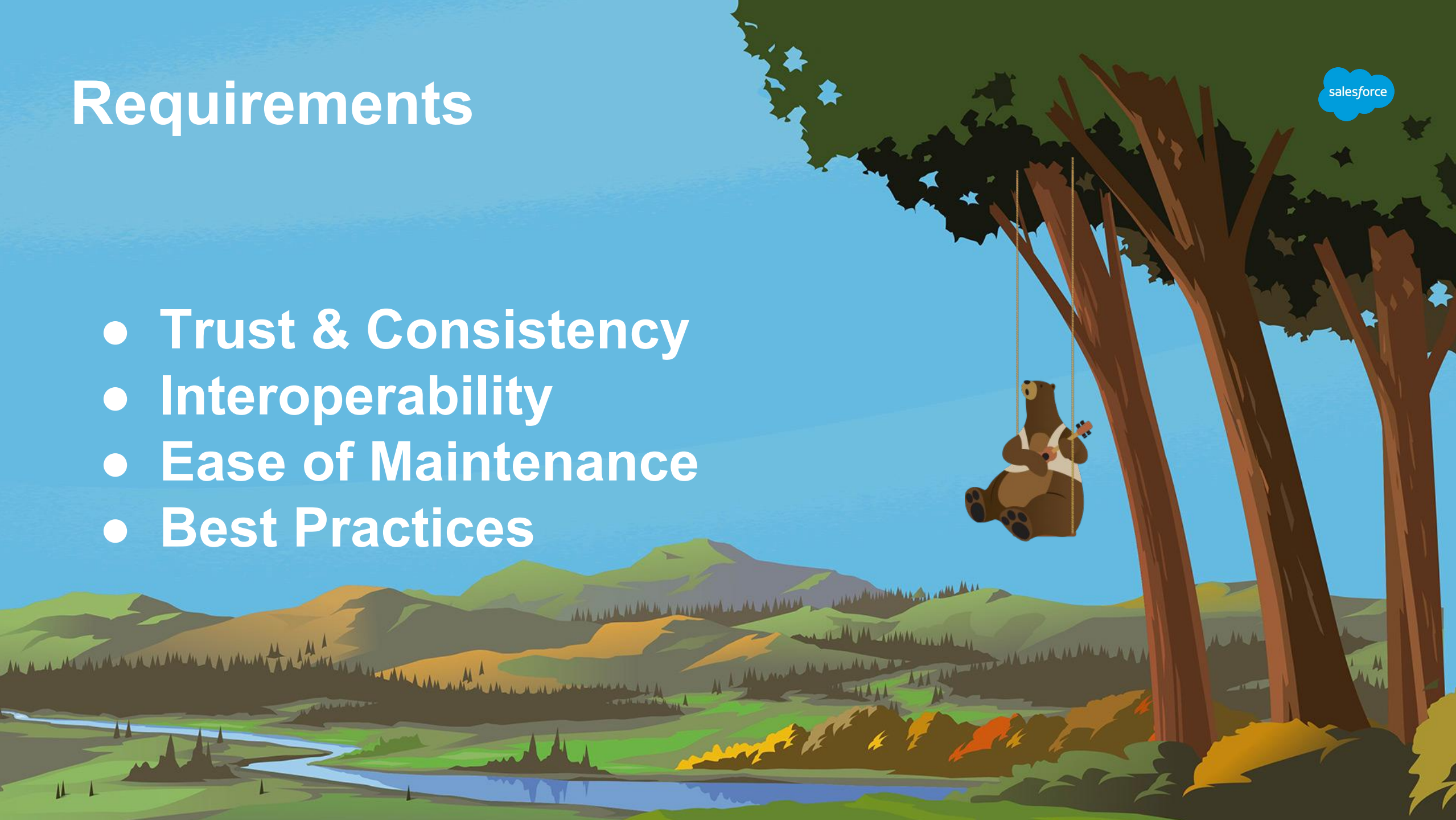


Chapter 1: Requirements & Challenges

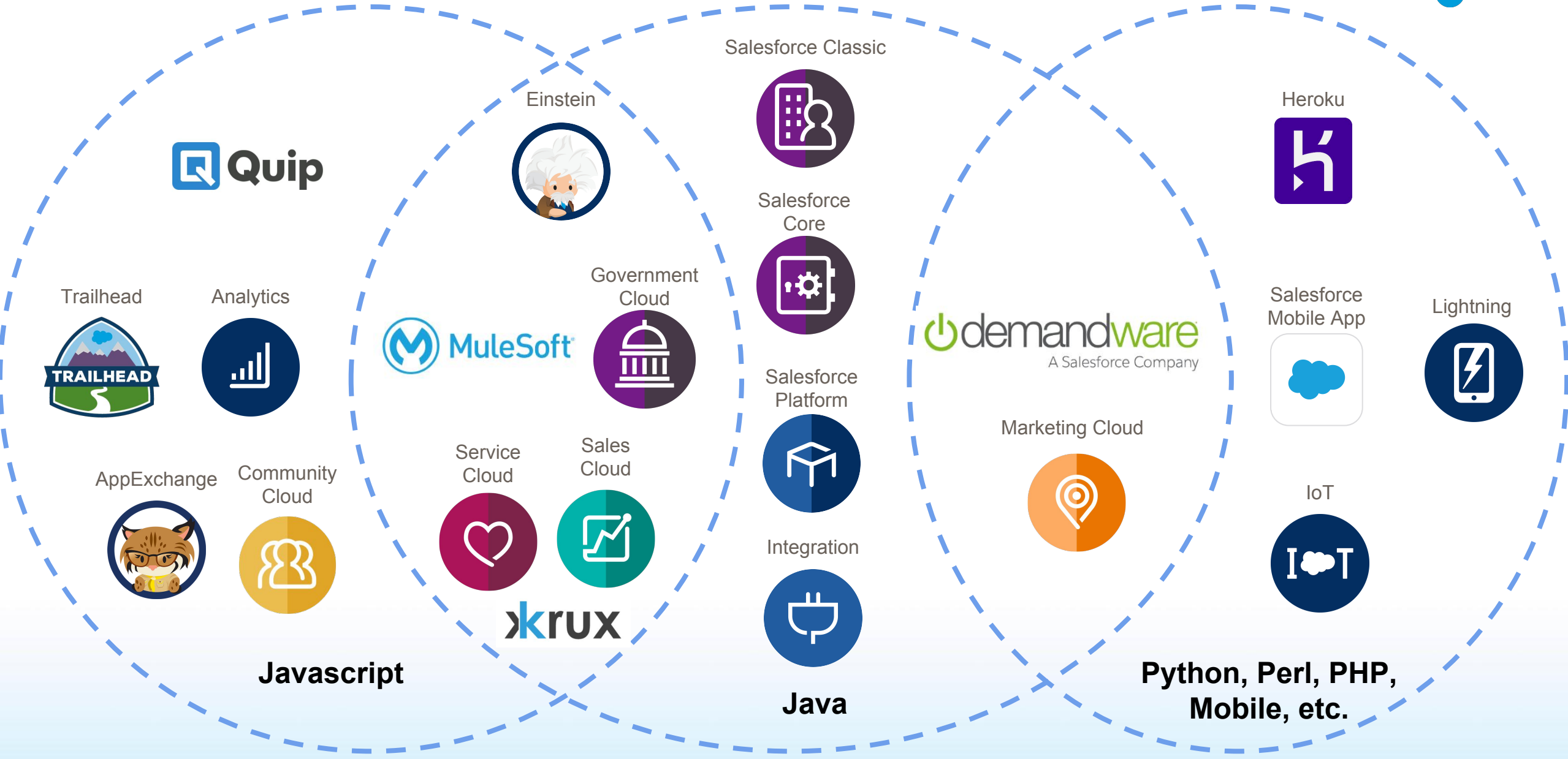
Requirements
+
Challenges

Requirements

- Trust & Consistency
- Interoperability
- Ease of Maintenance
- Best Practices



Challenges - Interoperability



Challenges - Complex Landscape

Technical Challenges

Cost of maintenance
210+ Locales to support
Concurrent support of ICU and JDK locale data

Incorrect/inconsistent Locale Data
Mixture of locale data from JDK, and ICU4J/CLDR

Multiple Tech Stacks
Legacy Code and Acquisitions

Organizational Challenges

Release Timing
Managing both internal and external customers

Large Feature Base
With hundreds of teams

Cross-Cloud Initiative
Existing products, new products and acquisitions

CORE ENGR
salesforce

Globalization

Sales Cloud | Service Cloud | Marketing Cloud | Commerce Cloud | Community Cloud | Salesforce Platform | Industries Cloud | Integration | Einstein

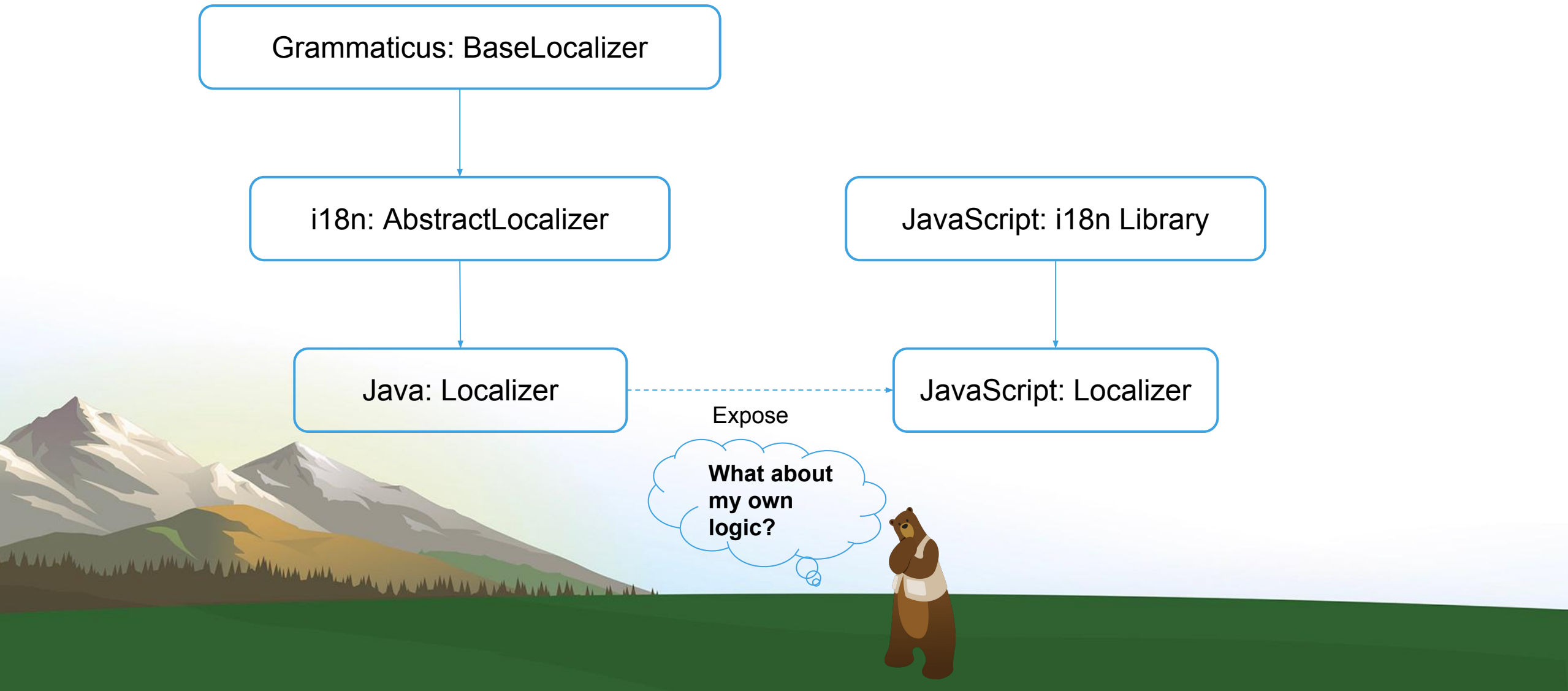
Our Customers

The Road to ICU

Chapter 2: Planning & Implementation

Planning
+
Implementation

Planning the Route



Planning the Route - Details

Attribute	Description	Sample Value
currency	The currency symbol.	"$"
currencyCode	The ISO 4217 representation of the currency code.	“USD"
decimal	The decimal separator.	"."
grouping	The grouping separator.	","
numberformat	The number format pattern.	"#,##0.###"
datetimeFormat	The date time medium format.	"MMM d, yyyy h:mm:ss a"
...		

Lightning Components $Locale
https://developer.salesforce.com/docs/atlas.en-us.lightning.meta/lightning/expr_locale_value_provider.htm

Planning the Route - Findings

- Accuracy 😓
- Consistency & Interoperability 😊
- Ease of Maintenance 😓

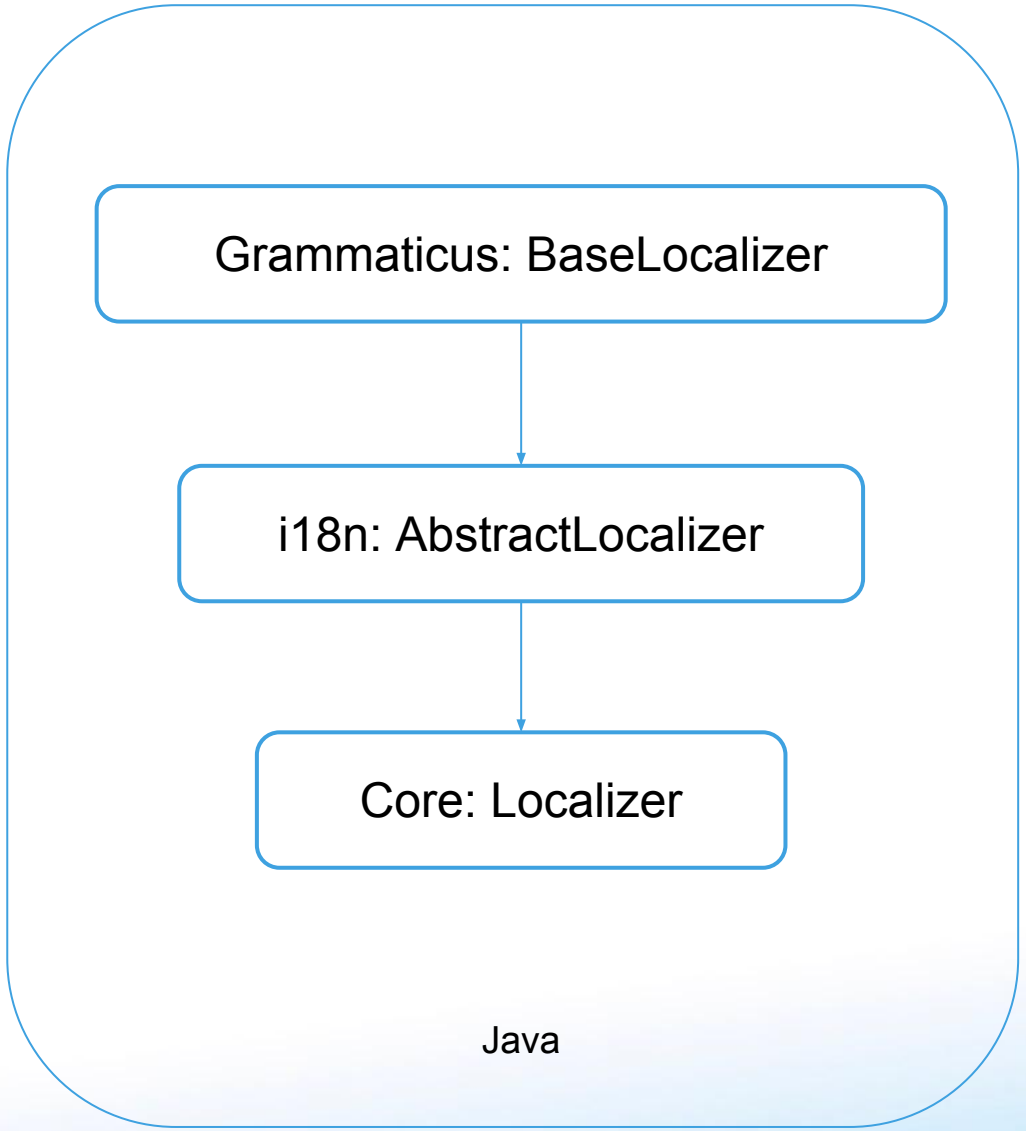
Problems:

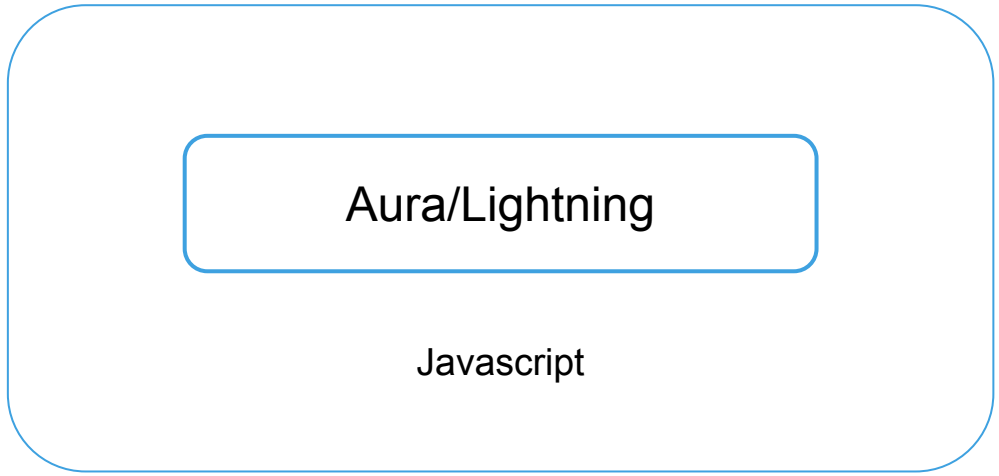
1. 210 locales with 10-20 patterns
2. Takes ~80% of the application bundle, big performance hit
3. Difficult to format with the patterns in Javascript

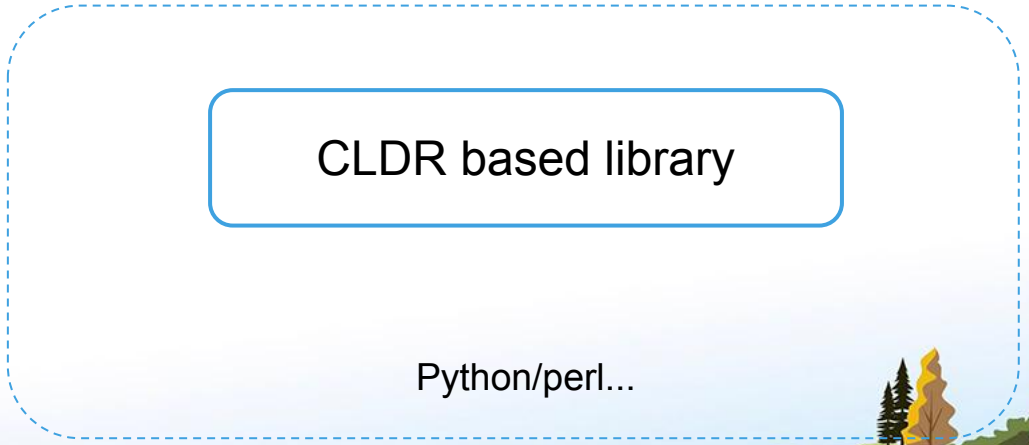
Example:

‘MM/dd/y HH:mm a’

Implementation: CLDR compatible libraries design

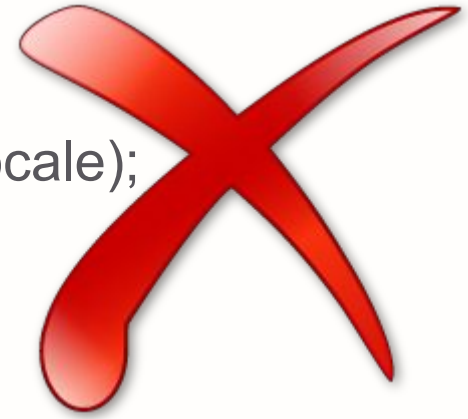






Implementation: Migrate to ICU4J

- Switch to use ICU4J APIs to retrieve locale data
  - com.ibm.icu.text.DateFormat df =
    com.ibm.icu.text.DateFormat.getDateTimeInstance(style, style, locale);

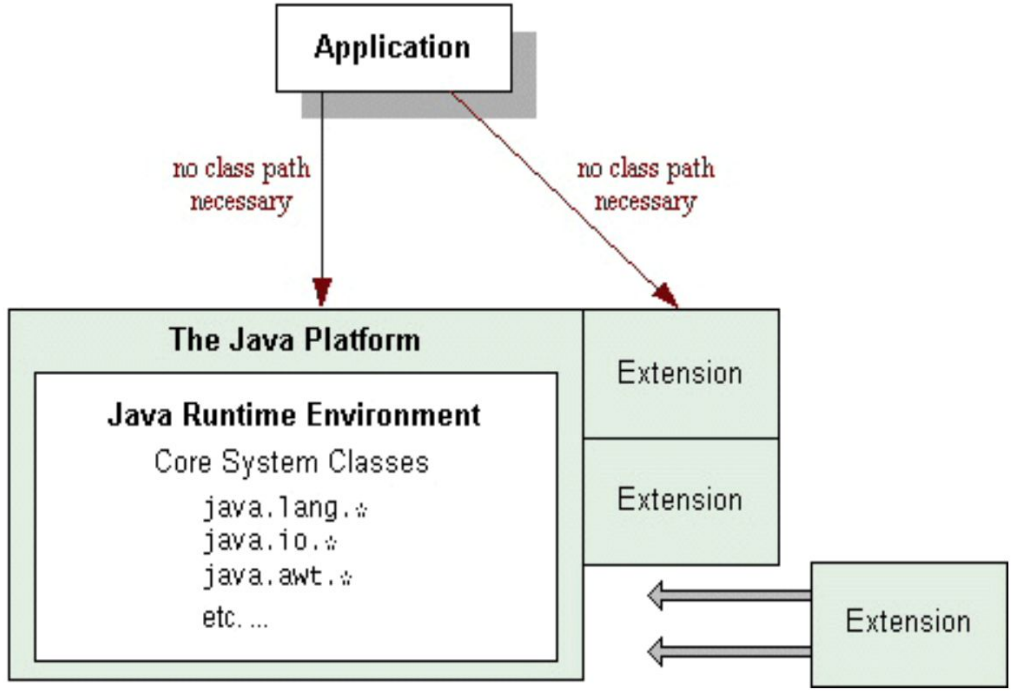


- JDK classes vs ICU classes
  - java.text.DateFormat -> java.text.format
  - com.ibm.icu.text.DateFormat -> com.ibm.icu.text.UFormat ->java.text.format
- 8772 DateFormat API calls in code base!

Implementation: SPI

Service Provider Interface (SPI)

java.util.spi	java.text.spi
• CurrencyNameProvider • LocaleServiceProvider • TimeZoneNameProvider • CalendarDataProvider	• BreakIteratorProvider • CollatorProvider • DateFormatProvider • DateFormatSymbolsProvider • DecimalFormatSymbolsProvider • NumberFormatProvider




```
java -Djava.ext.dirs=$JAVA_HOME/lib/ext:$ICU_SPI_DIR <your_java_app>
```

Implementation: choose locale data provider

- Set provider, java.locale.providers=COMPAT,SPI,CLDR
  - COMPAT : Java locale data
  - SPI : Service provider data
  - CLDR : bundled CLDR data

- Support both JDK and ICU locale data
  - Adapter to wrap ICU4J classes
  - Predicate to retrieve data

```
NumberFormatICU.wrap(com.ibm.icu.text.NumberFormat nf =
        com.ibm.icu.text.NumberFormat.getNumberInstance(locale))
```

Implementation: Adjusting to JDK9

JDK9

Changes:

- Removed extension and endorsed mechanism.
- Make CLDR as default locale provider.

`java.endorsed.dirs`

`lib/endorsed`

Solution:

- JDK-8062588 load providers for SPIs with the application class loader
- Transfer ICU jars into upgradable modules.
- Legacy locale data support.
  - Set java.locale.providers=COMPAT,SPI,CLDR

Implementation: Demo

1. Setup locale data service provider
2. Configure service provider
3. Concurrent support of JDK and ICU locale data

Implementation: Intl

`Intl`: the namespace for the ECMAScript Internationalization API

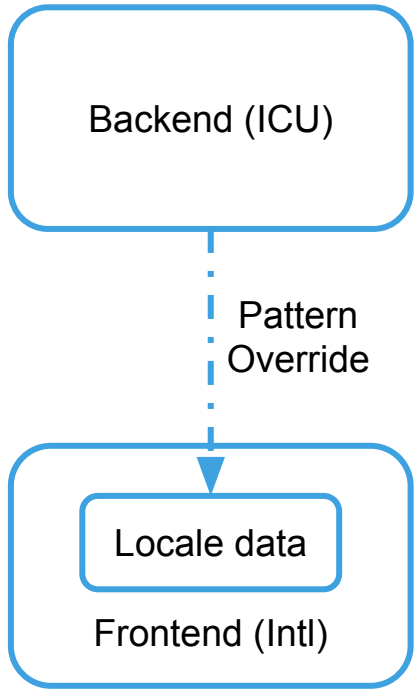
- Intl.Collator
- Intl.DateTimeFormat
- Intl.NumberFormat
- Intl.PluralRules

```
console.log(new Intl.DateTimeFormat('en-US').format(date));
// expected output: "12/20/2012"

console.log(new Intl.DateTimeFormat('en-CA').format(date));
// expected output: "2012-12-19"
```

Implementation: browser locale data consistency

Browser	Date	Time	Date & Time
Chrome-57	58%	47%	93%
Chrome-64	78%	70%	95%
Firefox-52	92%	83%	98%
Firefox-56	93%	83%	98%
Firefox-58	93%	83%	98%
MicrosoftEdge-13	70%	32%	79%
MicrosoftEdge-15	73%	34%	82%
MicrosoftEdge-16	74%	34%	83%
Safari-10	86%	65%	89%
Safari-11	88%	71%	88%
Internet explorer-11	39%	16%	74%
Nodejs-8.x	93%	84%	98%



Implementation: frontend library

```
module.exports = function formatDate(value, pattern, locale) {
  switch (locale) {
      case 'xx-XX':
          result = require('./exceptions/xx-XX')(value, pattern); break;
      case 'en-CA':
          result = require('./exceptions/en-CA')(value, pattern); break;
      default:
          var patterns = cache[locale];
          var dtf = patterns && patterns[pattern];
          if (!dtf) {
              dtf = new Intl.DateTimeFormat(locale, patternToOptions(pattern));
              cache[locale] = cache[locale] || Object.create(null);
              cache[locale][pattern] = dtf;
          }
          result = dtf.format(value);
  }
```

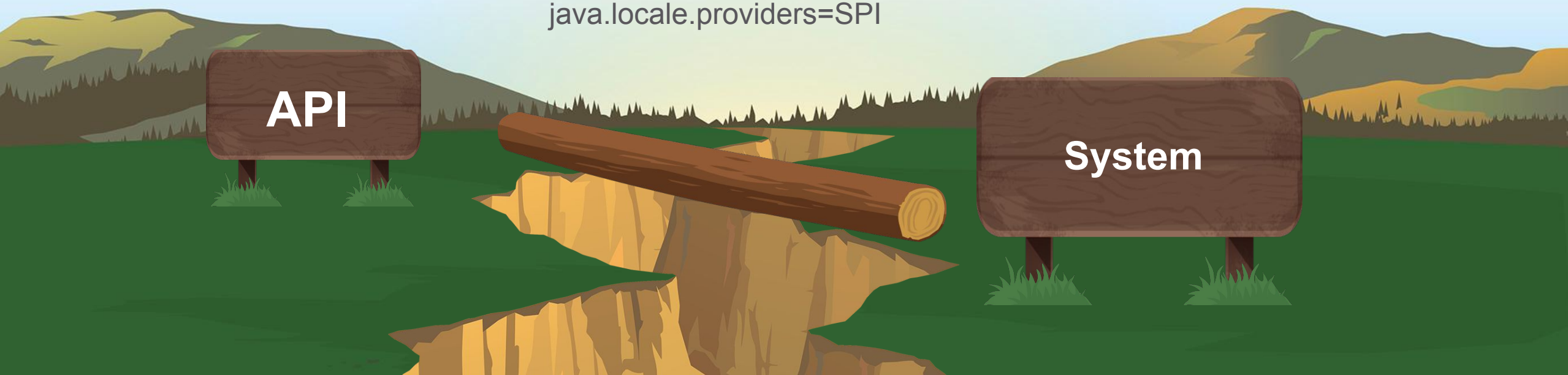
Implementation: Automation

- Concurrent support for both ICU4J and JDK locale data
- Benchmark build + automation (discovered 2K+ test failures)

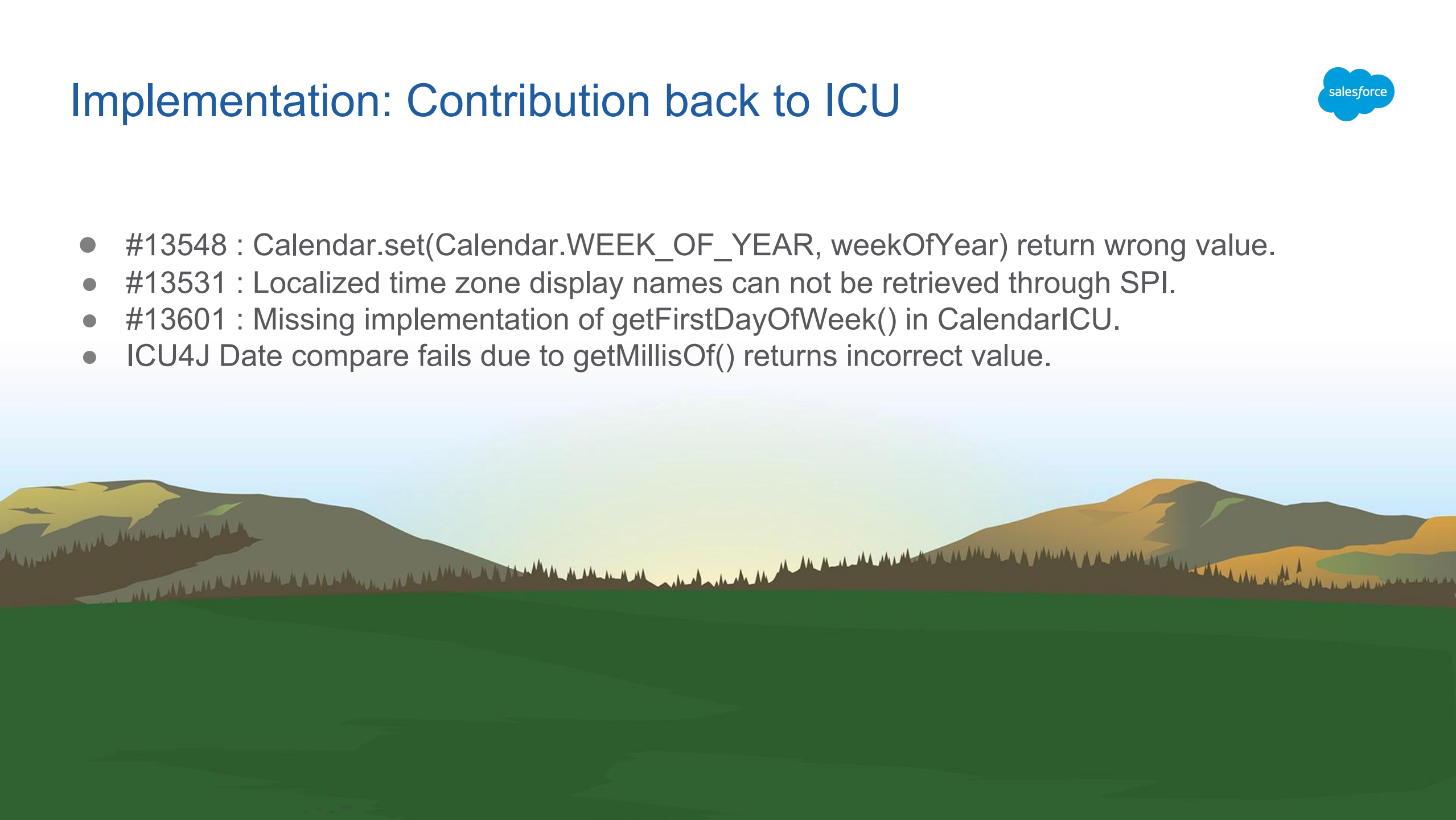
Implementation: Bridging the API and system gap

Phase 1: Consolidate APIs to use CLDR based libraries.

Phase 2: Make existing functional code compatible with CLDR without major revision in product code.



Implementation: Contribution back to ICU



- #13548 : Calendar.set(Calendar.WEEK_OF_YEAR, weekOfYear) return wrong value.
- #13531 : Localized time zone display names can not be retrieved through SPI.
- #13601 : Missing implementation of getFirstDayOfWeek() in CalendarICU.
- ICU4J Date compare fails due to getMillisOf() returns incorrect value.

Chapter 3: Case Studies

Case Studies

Case study #1 - Type Casting Exception

Cast without type check.

JDK-8190278 ClassCastException is thrown by java.util.Scanner when a NumberFormatProvider is used.

DecimalFormat df = (DecimalFormat) NumberFormat.getNumberInstance(locale);

Case study #2 - Differences in currency format in JDK and ICU

JDK:

- NUMBERSTYLE
- **CURRENCYSTYLE**
- PERCENTSTYLE
- SCIENTIFICSTYLE
- INTEGERSTYLE

($123,456.79)

ICU:

- ACCOUNTINGCURRENCYSTYLE ***($123,456.79)***
- CASHCURRENCYSTYLE
- **CURRENCYSTYLE**
- ISOCURRENCYSTYLE
- NUMBERSTYLE
- PERCENTSTYLE
- SCIENTIFICSTYLE
- INTEGERSTYLE
- PLURALCURRENCYSTYLE

- $123,456.79

Case study #3 - Pattern recognition

my_MM pattern in CLDR

A pattern generated from icu, dd-MM-yyyy B H:mm. The pattern 'B' cannot be recognized by JDK, which is causing exception for following code,

```
new SimpleDateFormat(pattern, locale)
```

Case study #4 - Inconsistent currency code

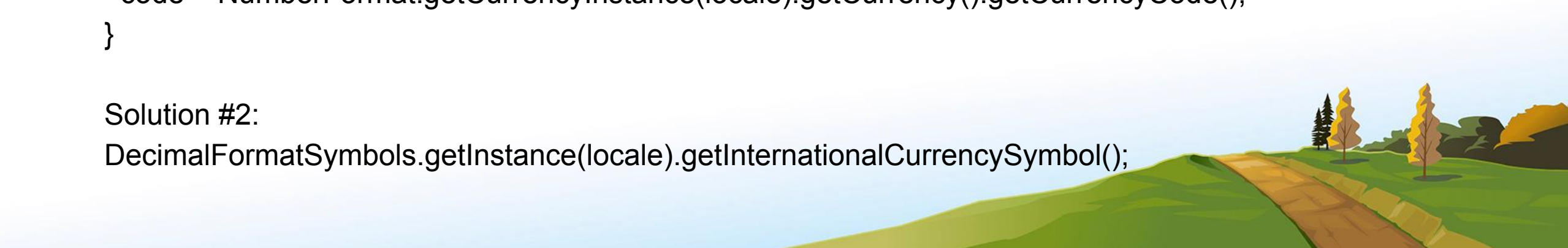
Exception for 'be_BY' and 'pt_ST' locales.

NumberFormat.getCurrencyInstance(locale).getCurrency().getCurrencyCode()

Solution #1: override the currency code for these two locales before calling getCurrency() method.

```
if (locale.toString().equals("be_BY")) {
  code = "BYR"; // JDK:BYR; ICU:BYN
} else if (locale.toString().equals("pt_ST")) {
  code = "STD"; // JDK: STD; ICU:STN
} else {
  code = NumberFormat.getCurrencyInstance(locale).getCurrency().getCurrencyCode();
}
```

Solution #2:
DecimalFormatSymbols.getInstance(locale).getInternationalCurrencySymbol();



Summary

- Accuracy, consistency and easy of maintenance.
- CLDR based libraries.
- Service provider.
- Plan ahead. Know the impact - and where


THANK YOU

Appendix

- CLDR
  http://cldr.unicode.org
- Locale-Sensitive Services SPI
  https://docs.oracle.com/javase/tutorial/i18n/locale/services.html
- ICU4J Locale Service Provider
  http://userguide.icu-project.org/icu4j-locale-service-provider
- Grammaticus
  https://github.com/salesforce/grammaticus