

Boost.Locale



Localization for the C++ World

Artyom Beilis



What is Boost.Locale?

- Developed for CppCMS
- Adjusted for Boost standards
- April 2011: Accepted into Boost

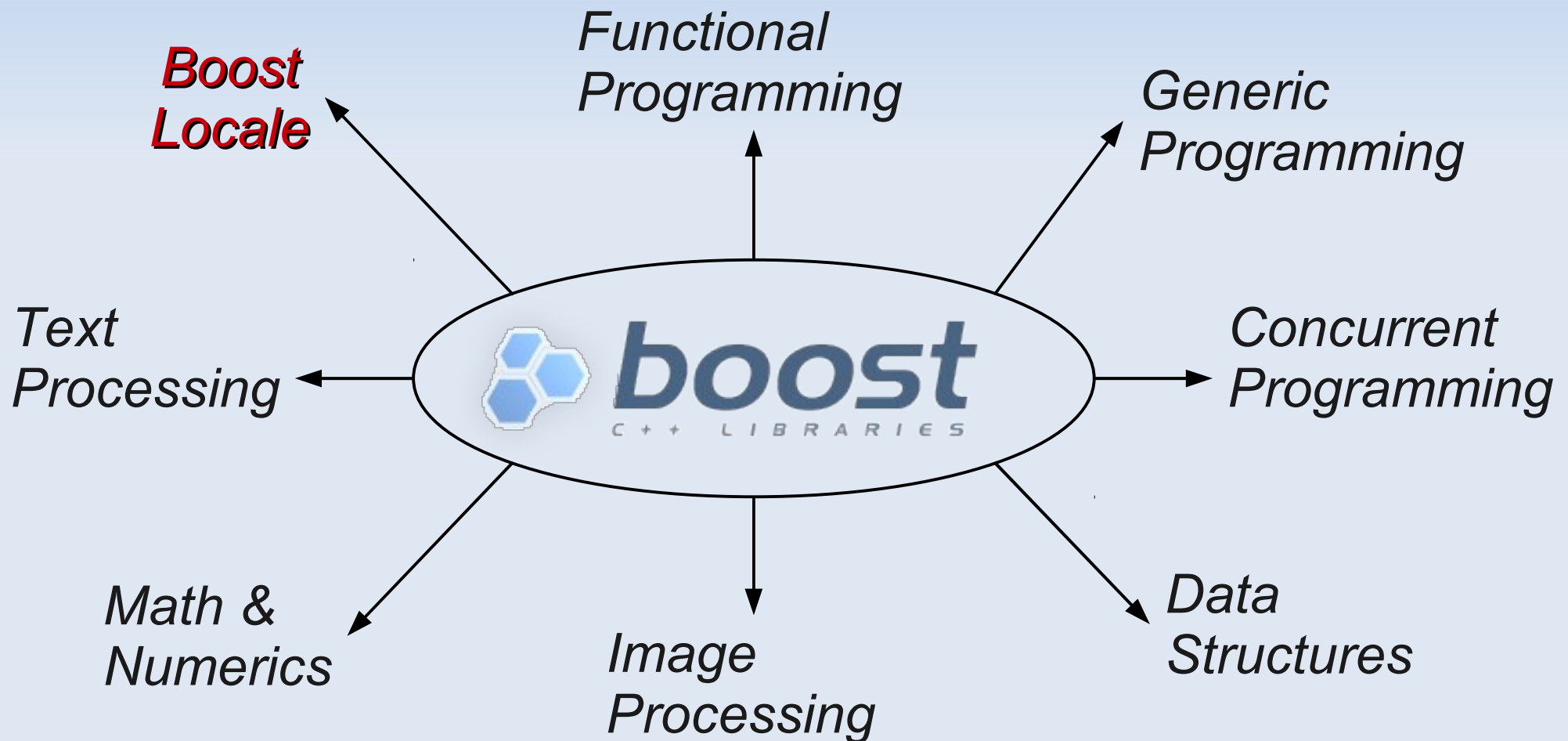


Boost.Locale

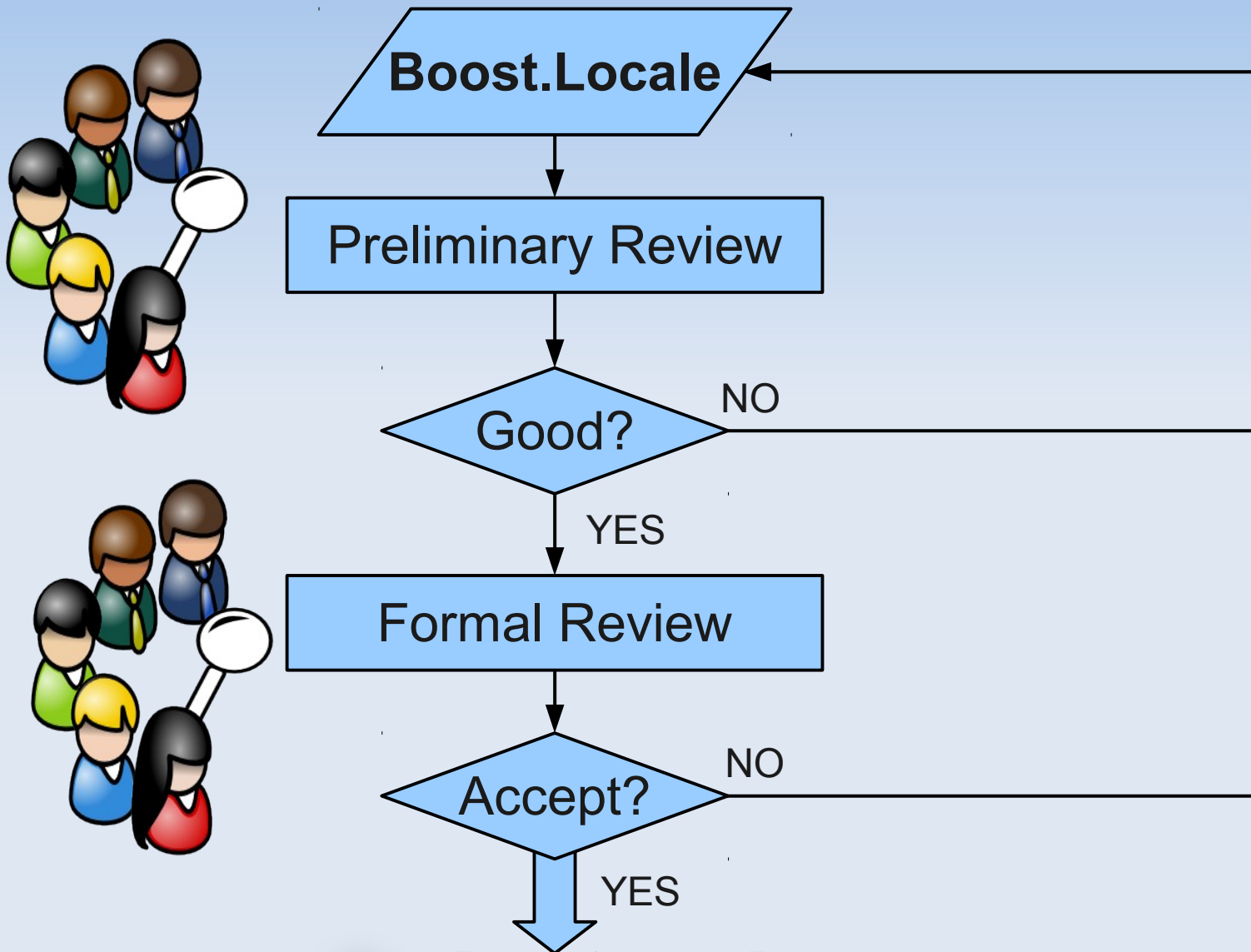
What is Boost?



Extends the Standard C++ Library



Formal Review



Influences The Standard



Boost

boost::shared_ptr

boost::regex

boost::function

boost::bind

boost::tuple

boost::unordered_map



C++0x

std::shared_ptr

std::regex

std::function

std::bind

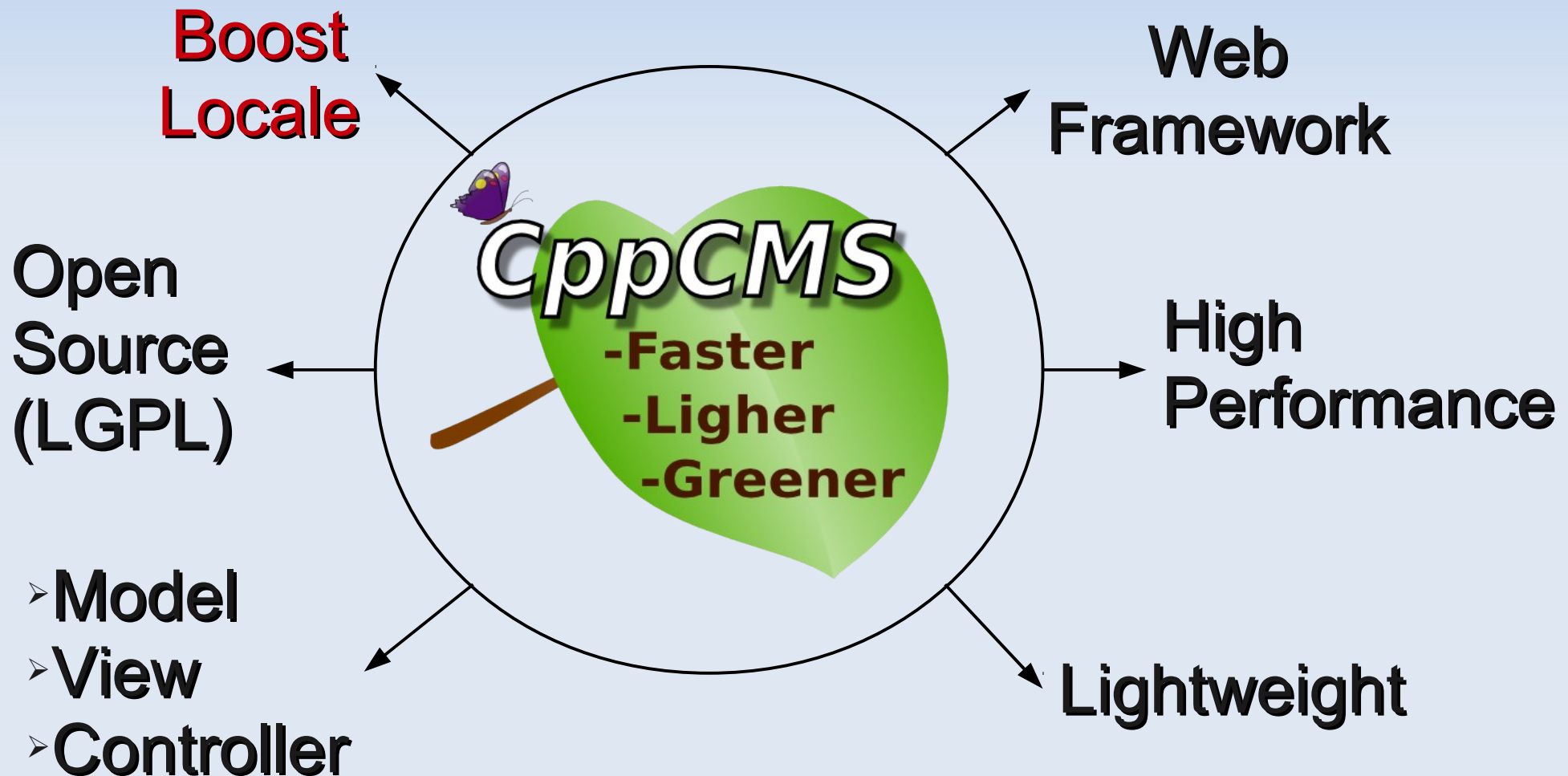
std::tuple

std::unordered_map

What is CppCMS?



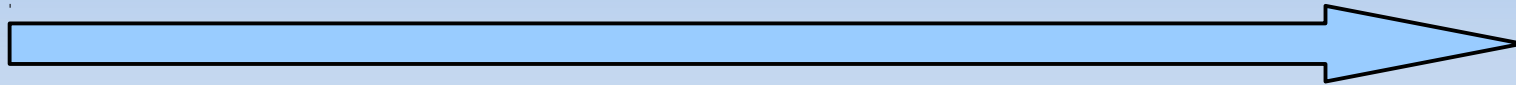
C++ Web Development Framework









CppCMS – Web Framework



Performance



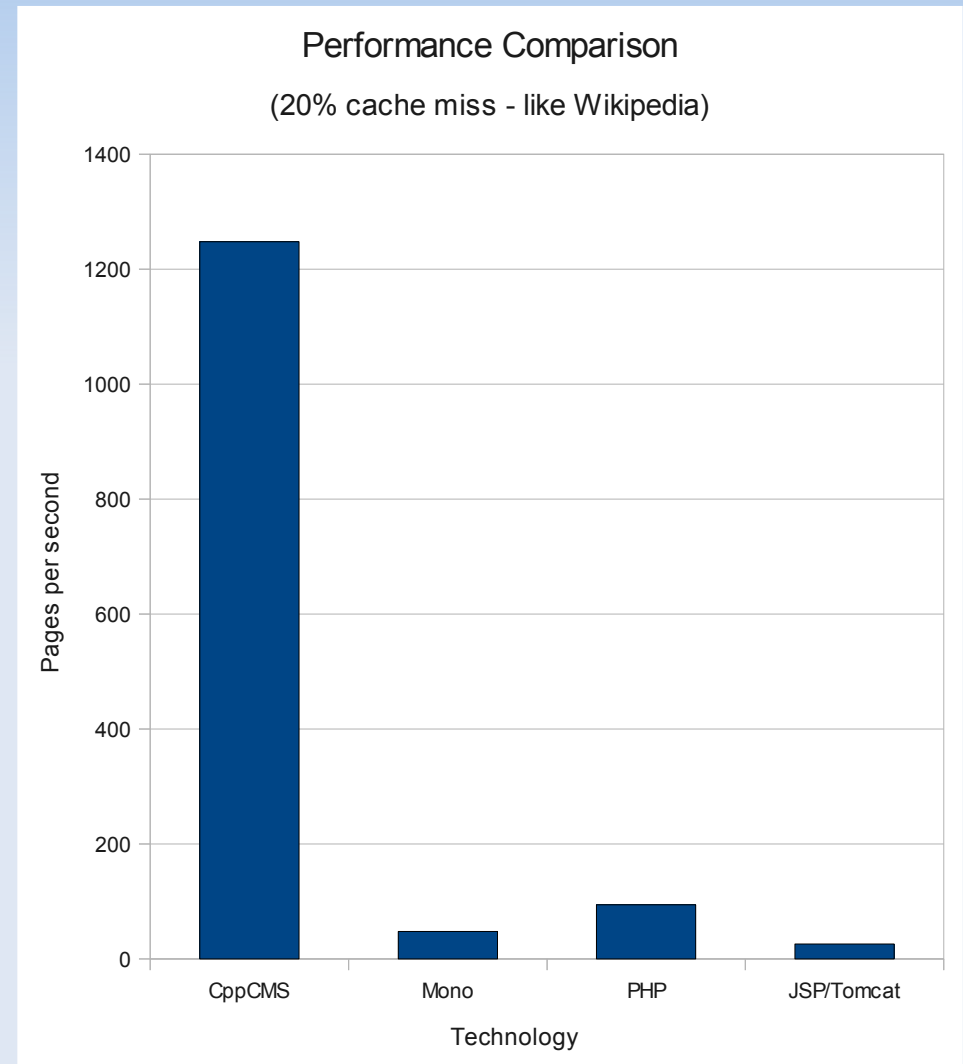
Bytecode	Just-in-Time	Native
PHP 	Java 	C++ 
Ruby 	C# 	
Python 		

CppCMS – High Performance



Case Study:

- Leading technologies:
 - CppCMS
 - Asp.Net / Mono
 - PHP5
 - JSP / Tomcat
- MySQL database
- Markdown text format
- 20% cache miss (like Wikipedia)



CppCMS – High Performance



What Users Tell?

- **Allan Simon – Tatoeba.org:**

*for the moment average page generation: 400 μ s,
1000 times faster...*

- **Dhiti - a content discovery engine:**

*We chose CppCMS since we wanted **high performance** on a C++ based real time discovery engine. We've been happy with CppCMS and look forward to its evolution as a great development framework.*

- Goals & Requirements
- Features
- Change the standards?

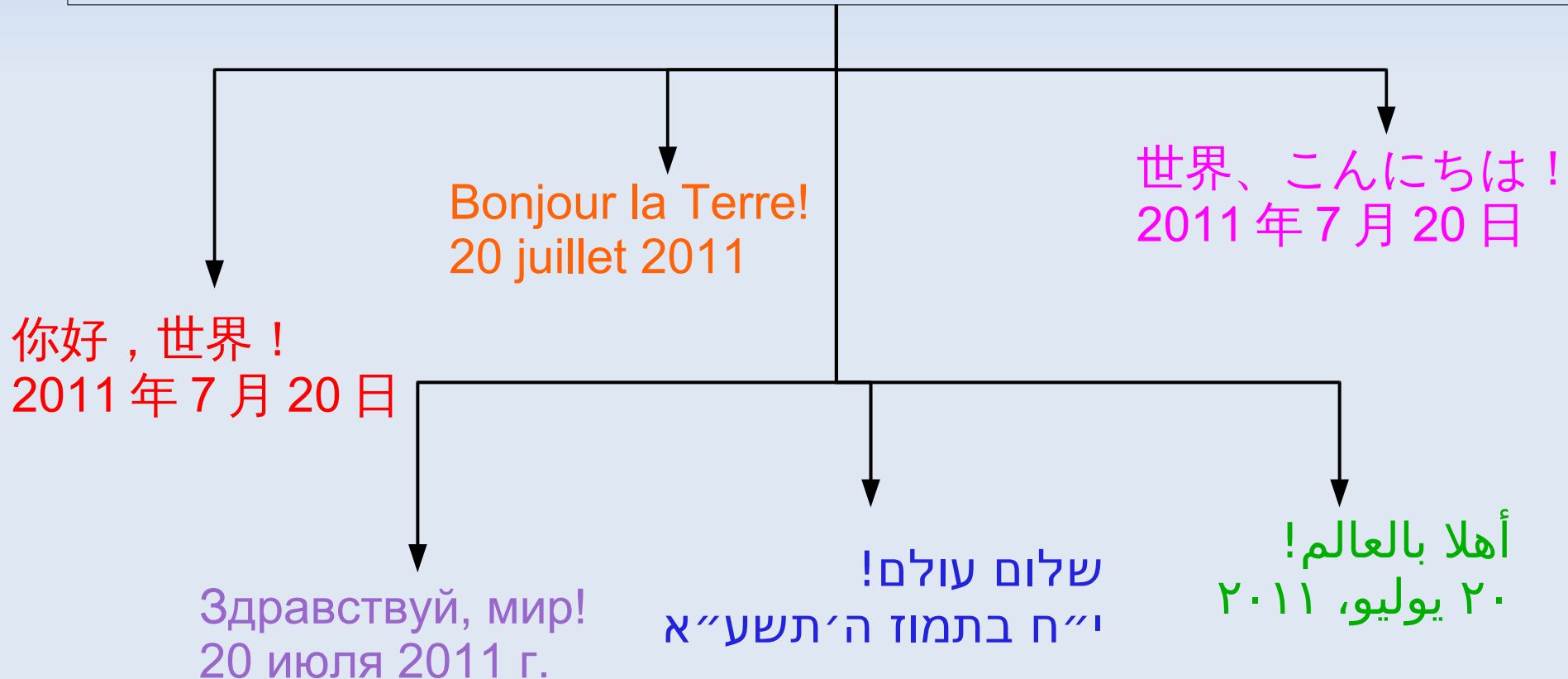


Boost.Locale

Make it Simple

你好，世界！

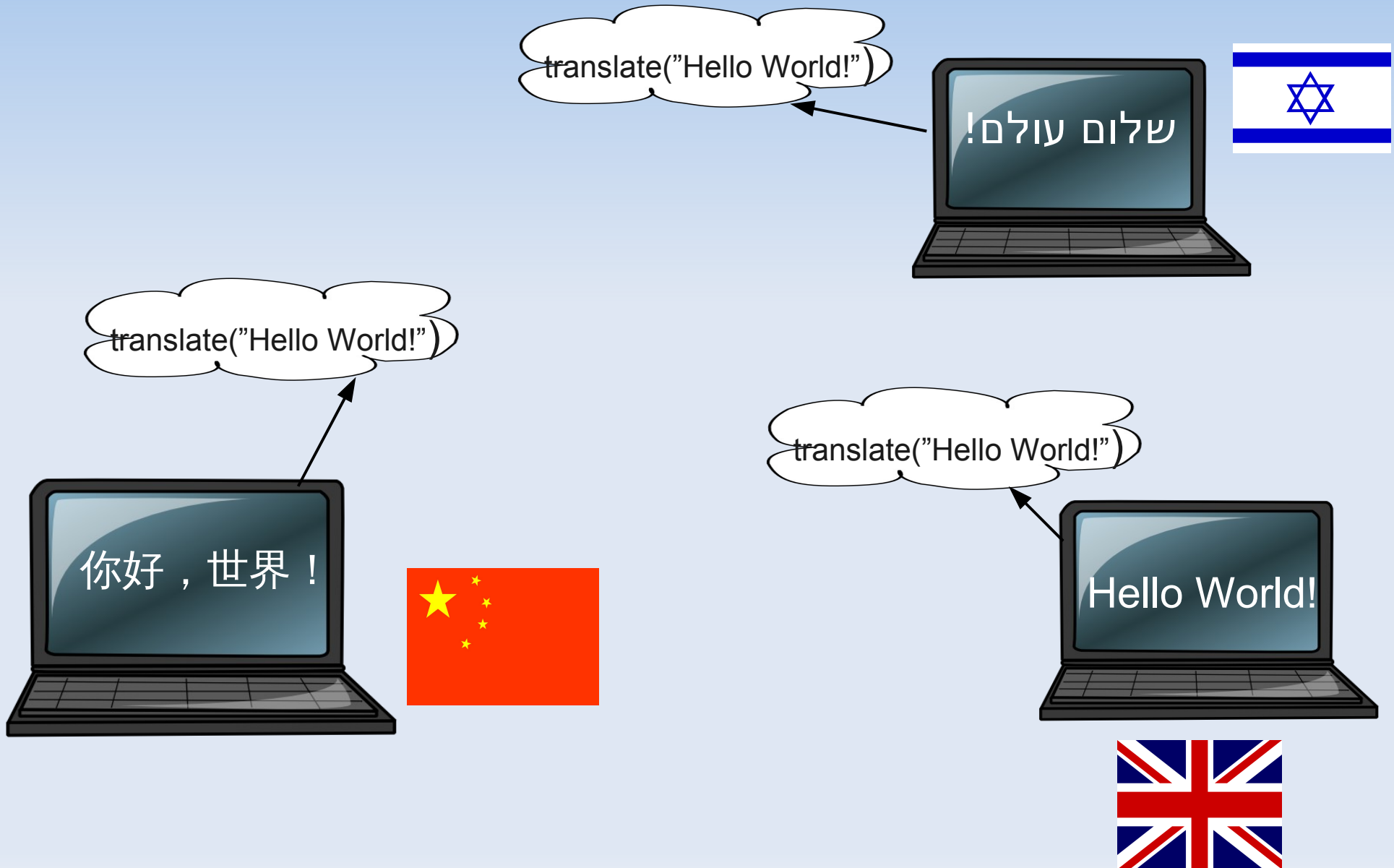
```
void my_localized_function()  
{  
    std::cout << translate("Hello World!") << std::endl;  
    std::cout << as::date << std::time(0) << std::endl;  
}
```



- Message translation: Hello World → שלום עולם
- Collation (sorting in alphabetical order)
- Values Formatting:
 - When is 3/4/2011? (April 3rd or March 4th)
 - How much is 1,234? (1234 or $1\frac{234}{1000}$)
- Boundary analysis:
 - Find Words: 生きるか死ぬか、それが問題だ。
 - Find Letters: "שלום"
- And much more...

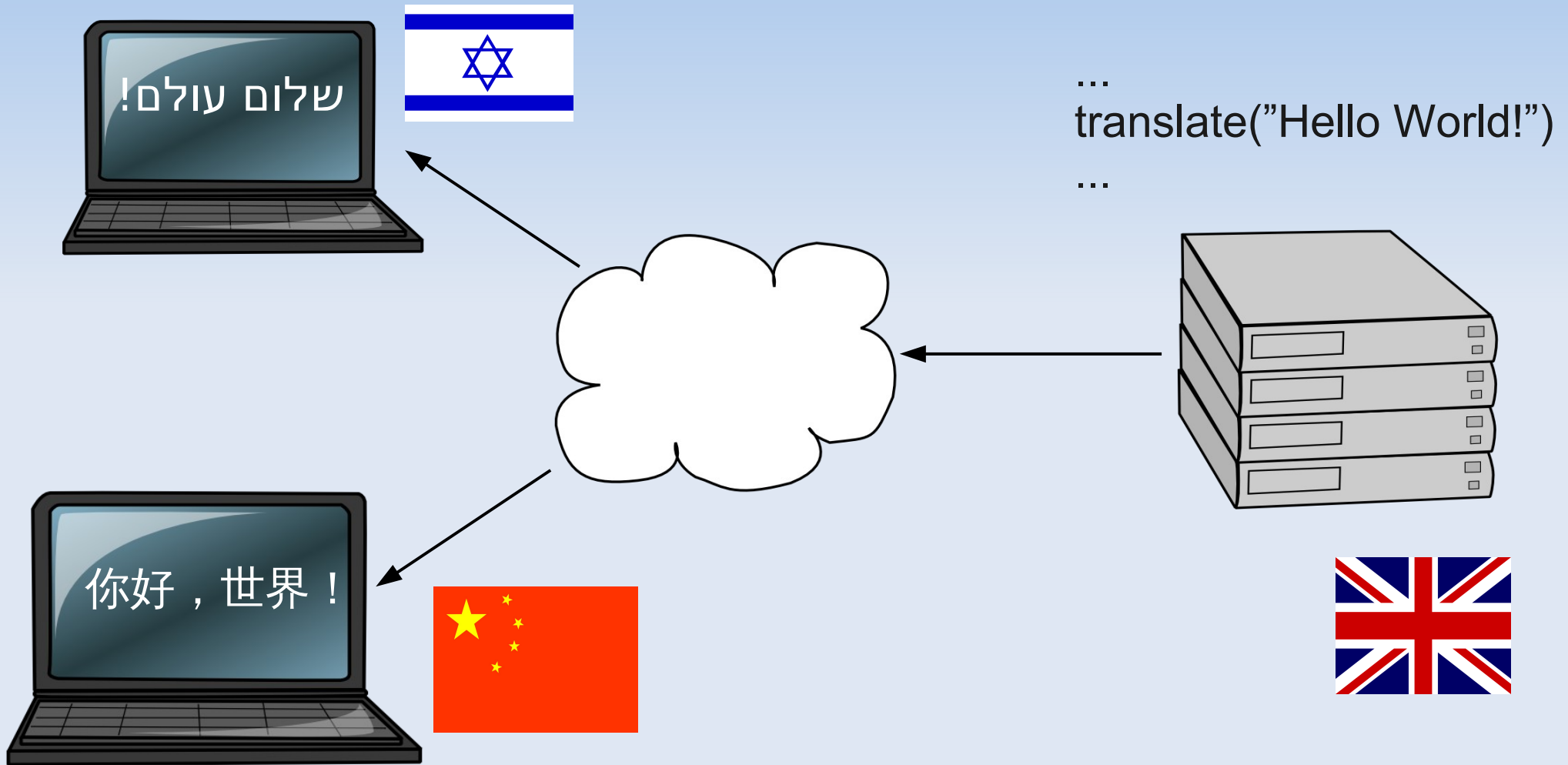
GUI Oriented

Bonjour la Terre!



Service Oriented

أهلا بالعالم!



Standards Friendly

שלום עולם!

Yes

std::string

char

std::locale

std::ostream

No

QChar

QString

tchar

UnicodeString

UChar

CString

ustring

Library Features

世界、こんにちは！

Tutorials: http://cppcms.sf.net/boost_locale/html/

- Message Formatting (string translation)
- Operations with dates, time, timezone for multiple calendars
- Multiple Levels Collation (sorting)
- Boundary analysis for characters, words, etc.
- Number and monetary formatting, spelling and parsing.
- Character set conversion.
- Unicode Normalization
- Case conversion
- Full UTF-8 support
- POSIX locale names (even on Windows)
- char, wchar_t, char16_t and char32_t support

Standard Names

你好，世界！

- Uniform POSIX Locale names on all platforms.
- UTF-8 is **default encoding** on Windows

UTF-8	ANSI Codepage
he_IL.UTF-8	Hebrew_Israel.1255
fr_FR.UTF-8	French_France.1252



GNU Gettext - Improved

GNU Gettext

- De-facto standard
- Plural Forms
- Context Support

Boost.Locale

- Multiple Locales
- Permissive License
- Wide characters

```
std::cout << translate("Hello World!") << std::endl;
```

- Transparent Support of non-Gregorian dates:
 - `he_IL.UTF-8` – Gregorian
 - `he_IL.UTF-8@calendar=hebrew` – Hebrew
- I/O streams integration
- Calendar manipulations

```
time_t time = std::time(0);
std::cout << "Local time:      " << as::time << time <<std::endl;
std::cout << "Universal time:" << as::gmt  << time <<std::endl;

date_time now;
now = period::sunday(); // Move to the Sunday of the local week
```

- Correct case handing: lower, upper, title
 - Not one-to-one: “grüßen” → “GRÜSSEN”
 - Context dependent: “ὈΔΥΣΣΕΎΣ” → “ὀδυσσεύς”
 - UTF-8 friendly

```
std::string upper = toupper("grüßen");  
std::string normalized = normalize(text, norm_nfd);
```

Improve the C++ standard

From the Boost.Locale review:

*This library is potentially **extremely** useful because it **leverages** an important chunk of standard C++ that is sadly underutilised merely due to a few design and implementation flaws.*

...

*This is a **very** strategic library...*

There is localization!



There is a built-in support of localization in C++

- Locale Container: `std::locale`
- Messages: `std::messages`
- Numbers: `std::num_put`
- Date/Time: `std::time_put`
- Collation: `std::collate`
- Conversions: `std::toupper`
`std::tolower`
- Analysis: `std::isalpha`
`std::isspace`

Each one of them has design flaws!

Goals



- Standardize locale handling across different compilers vendors
- Improve current standards that are linguistically broken
- Make Unicode and UTF-8 the default encoding for the C++ standard library

Questions?

Do you have any questions?

質問はおありですか。

Есть вопросы?

Маєте якісь питання?



יש לכם שאלות?

Avez-vous des questions ?

Copyrights

- Copyright © 2011 Artyom Beilis.
This Work is licensed under a Creative Commons Attribution 2.5 Israel License
- Sample sentences are taken from Tatoeba.org and licensed under Creative Commons Attribution License 2.0 (fr).