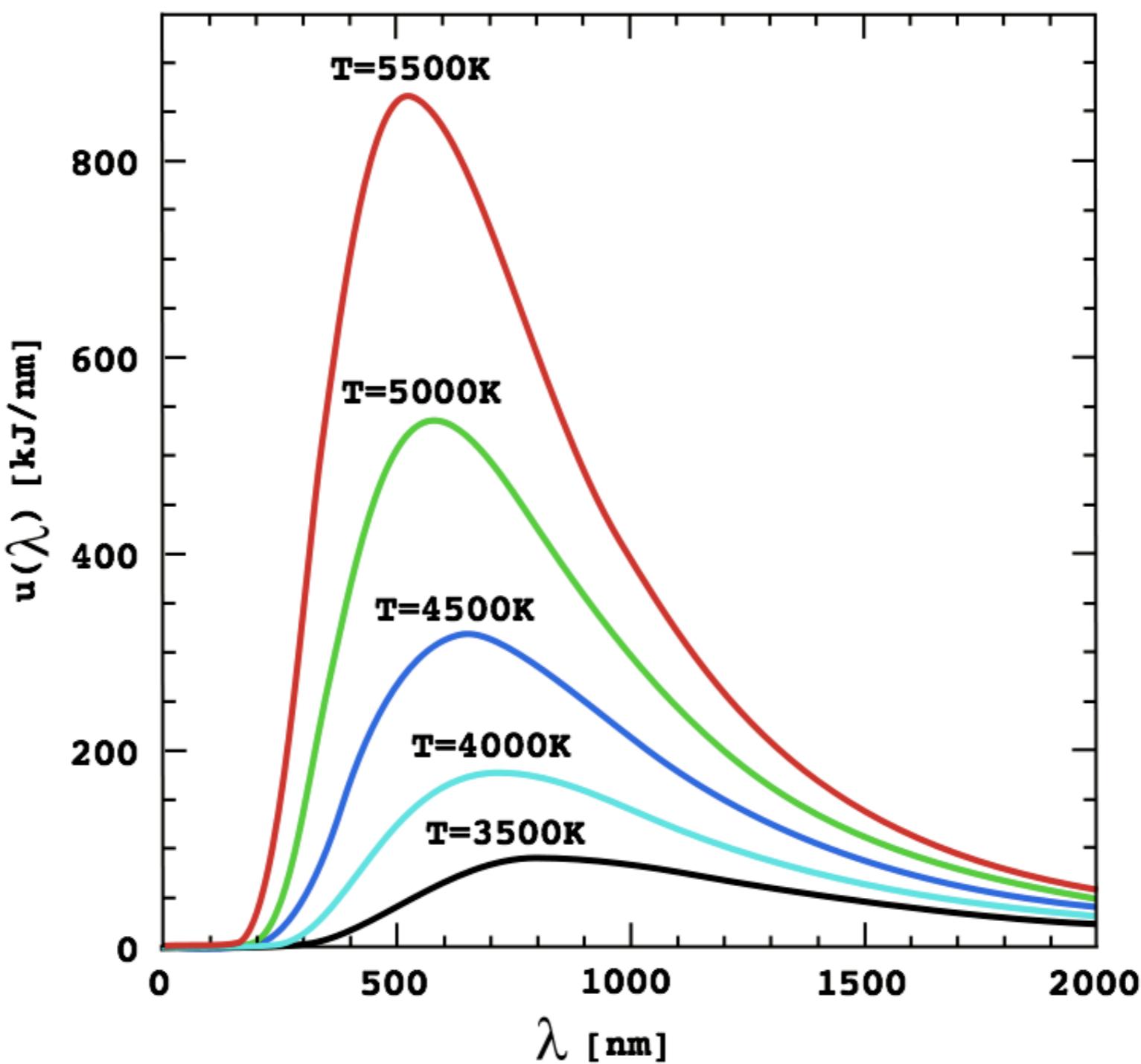




Andrei Zmievski
Chief Architect
Outspark, Inc

PHP for Grown-ups

How 5.3, 6, and intl will change your life



$$I(\lambda, T) = \frac{2hc^3}{\lambda^5} \frac{1}{e^{\frac{hc}{\lambda kT}} - 1}$$





PHP 6 = PHP 5 + Unicode



PHP 5 = PHP 6 - Unicode



Unicode = PHP 6 – PHP 5



What is PHP?

Ha. Ha.



What is Unicode?

and why do I need?

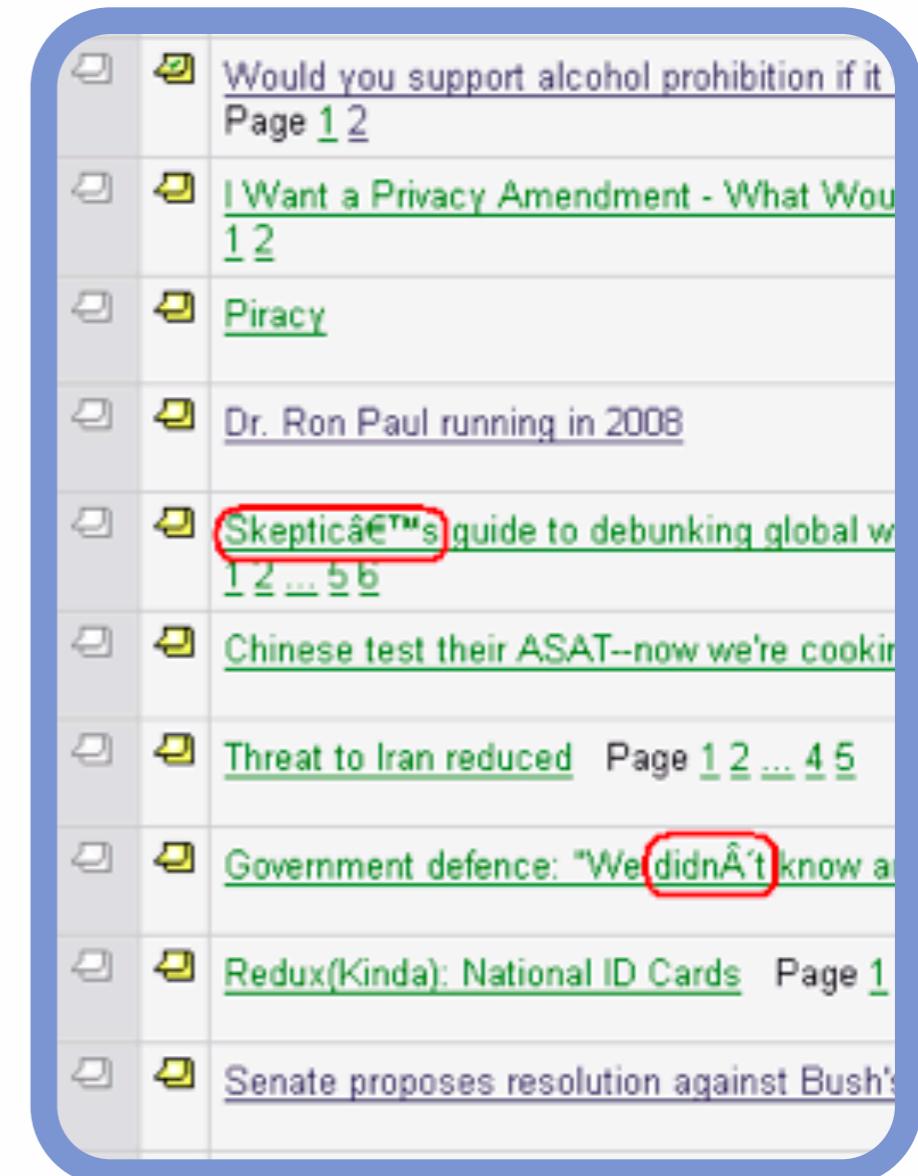


mojibake

もじばけ

mojibake

phenomenon of incorrect, unreadable characters shown when computer software fails to render a text correctly according to its associated character encoding



Subject:

From: [Eyal.Rozenberg <eyalroz@fastmail.fm>](mailto:Eyal.Rozenberg@fastmail.fm)

Date: 11:42

æåäé äôéñ÷ä äøàùåðä ùì ääääòä åäéà iëé:
áùôú-ëúéåä ùëéååðä iéíéí iùíàí.

æåäé äôéñ÷ä äùðééä åáä úåëíä iiöåä áí :
áùôú-ëúéåä ùëéååðä iùíàí iéíéí: äíéíä t
éù iééùø äääòä æå?

3

Unread: 0

mojibake

Unicode

provides a unique number
for every character:

no matter what the platform,

no matter what the program,

no matter what the language.



unicode standard

- Developed by the Unicode Consortium
- Covers all major living scripts
- Version 5.0 has 99,000+ characters
- Capacity for 1 million+ characters
- Widely supported by standards & industry

generative

- Composition can create “new” characters
- Base + non-spacing (combining) character(s)

A + ° = Å

U+0041 + U+030A = U+00C5

a + ^ + . = â

U+0061 + U+0302 + U+0323 = U+1EAD

a + _ + ^ = ā

U+0061 + U+0322 + U+030C = ?



unicode != i18n

- Unicode simplifies development
- Unicode does not fix all internationalization problems



definitions

Internationalization

I18n

To design and develop an application:

- ✓ without built-in cultural assumptions
- ✓ that is **efficient** to localize

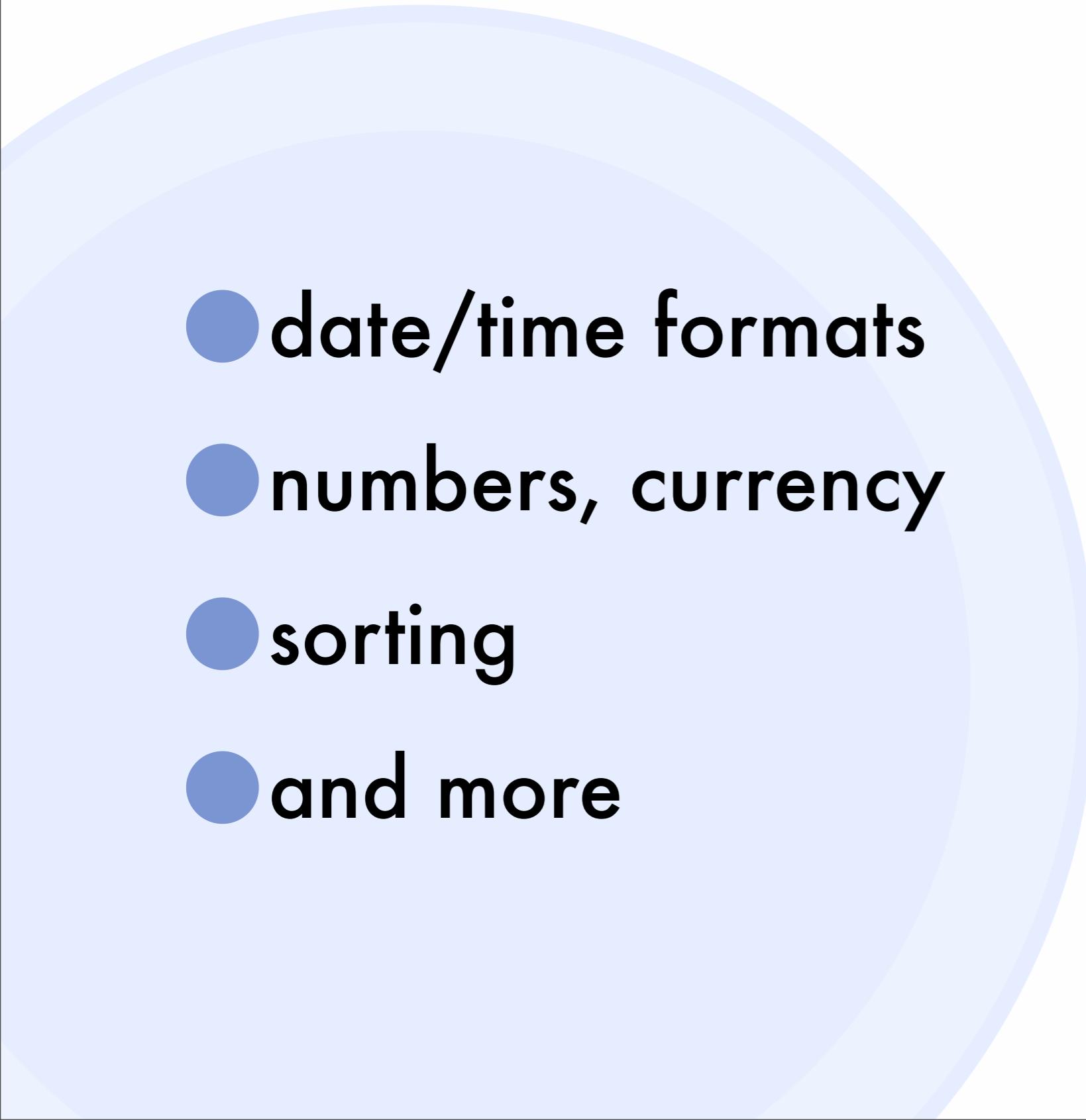
Localization

L10n

To tailor an application to meet the needs of a particular region, market, or culture



localized data

- 
- date/time formats
 - numbers, currency
 - sorting
 - and more

locale data



- I18N and L10N rely on consistent and correct locale data
- Problem with POSIX locales: not always consistent or correct

- Hosted by Unicode Consortium
- Goals:
 - Common, necessary software locale data for all world languages
 - Collect and maintain locale data
 - XML format for effective interchange
 - Freely available
- Latest release: July 2008 (CLDR 1.6)
- 374 locales, with 137 languages and 140 territories



PHP 6



unicode support

- Everywhere: engine, extensions, API
- Native and complete
- Uses industry-standard ICU library

Grab first 5 titles from Reuters China feed, clean up, and send out as JSON

```
$xml = simplexml_load_file(  
    'http://feeds.feedburner.com/reuters/CNTbusinessNews/');  
  
$titles = array();  
$i = 0;  
foreach ($xml->channel->item as $item) {  
    // each title looks like this: (台灣匯市) 台幣兌美元  
    $title = preg_replace('!\p{Ps}.*\p{Pe}\s*!', '', $item->title);  
    $titles[] = $title;  
    if (++$i == 5) break;  
}  
  
echo json_encode($titles);
```

string types



Unicode

- text

- default for literals, etc

Binary

- bytes

- everything \notin Unicode type

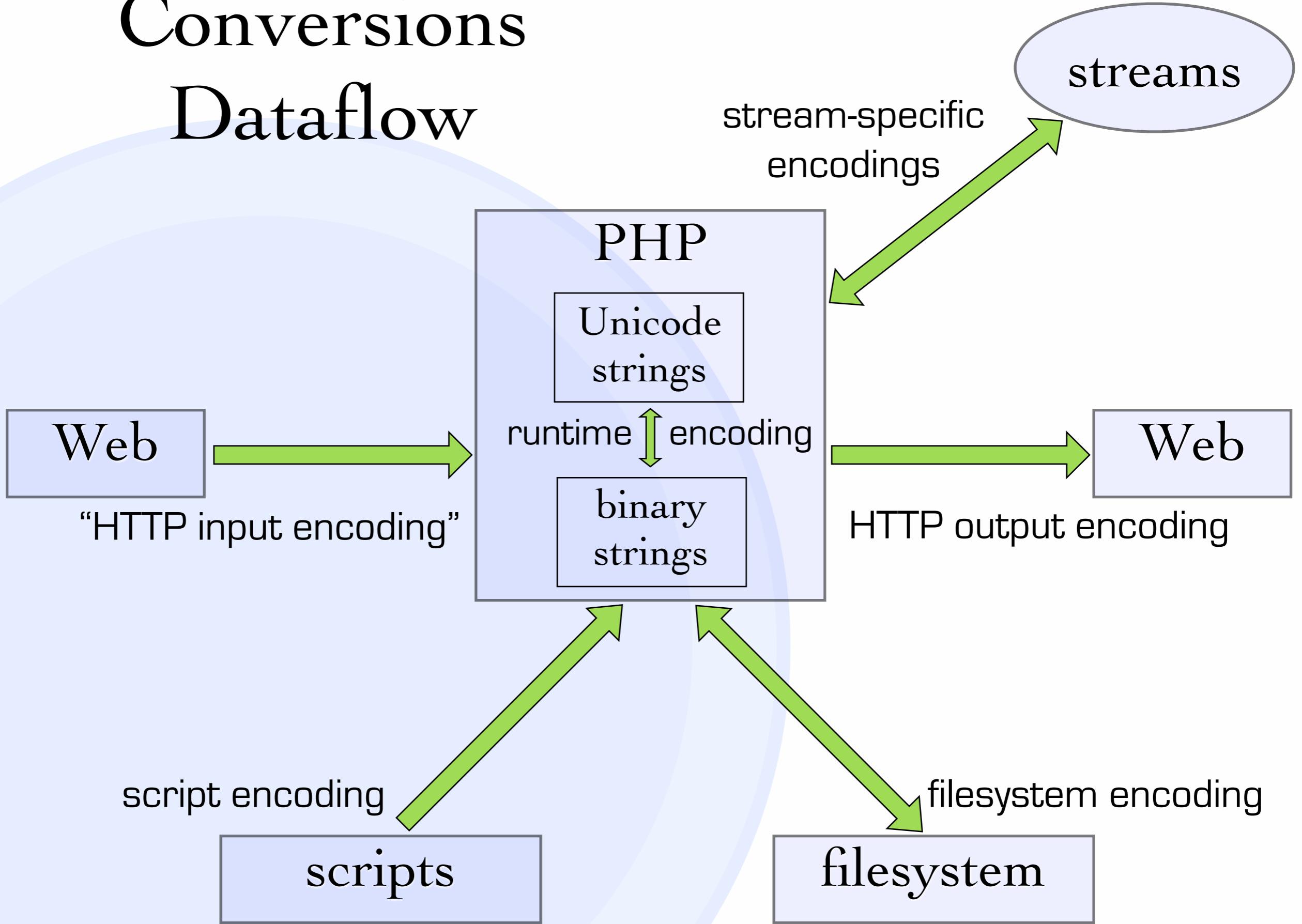
string types



- internal processing: Unicode
- interface to outside world: binary

Conversions

Dataflow





String literals are Unicode

```
$str = "Hello, world!";      // Unicode string
echo strlen($str);          // result is 13
```

```
$jp = "検索オプション";     // Unicode string
echo strlen($jp);           // result is 7
```

String offsets work on code points

```
$str = "大学";      // 2 code points
echo $str[1];      // result is 学
$str[0] = 'サ'; // full string is now サ学
```



Unicode identifiers are allowed

```
class コンポーネント {  
    function ಫುಂಕ್ಷನ್ { ... }  
    function சிவாஜி கணேசன் { ... }  
    function ଫୁନ୍କ୍ଷନ୍ { ... }  
}  
  
$プロバイダ = array();  
$プロバイダ['רְגִוְלָהּ'] = new コンポーネント();
```

functions



Functions understand Unicode text

- `strtoupper()` and friends do proper case mapping

```
$str = strtoupper("fußball"); // result is FUSSBALL
```

```
$str = strtolower("ΣΕΛΛΑΣ"); // result is σελλάς
```

- `strip_tags()` works on complex text

```
$str = strip_tags("雅<span>είναι</span>通");
```

- `strrev()` preserves combining sequences

```
$u = "Vi\u0302\u0323t Nam"; // Việt Nam
```

```
$str = strrev($u); // result is maN t̄eiV,  
// not maN t̄eiV
```



- Built-in support for converting between Unicode strings and other encodings on the fly
- Reading from a UTF-8 text file:

```
$fp = fopen('somefile.txt', 'rt');  
$str = fread($fp, 100); // returns 100 Unicode characters
```

- Writing to a UTF-8 text file:

```
$fp = fopen('somefile.txt', 'wt');  
fwrite($fp, $uni); // writes out data in UTF-8 encoding
```

Default encoding:

```
stream_default_encoding('Shift-JIS');
$data = file_get_contents('somefile.txt', FILE_TEXT);
// ... work on $data ...
file_put_contents('somefile.txt', $data, FILE_TEXT);
```

Custom contexts:

```
$ctx = stream_context_create(NULL,
                            array('encoding' => 'big5'));
$data = file_get_contents('somefile.txt', FILE_TEXT, $ctx);
// ... work on $data ...
file_put_contents('somefile.txt', $data, FILE_TEXT, $ctx);
```

text iterator



- Use it when iterating over text in a linear fashion (instead of [] operator)
- Iteration over code points, characters, graphemes, words, lines, and sentences forward and backward
- Also provides ICU's boundary analysis API

text iterator



Iterate over characters

```
$text = "nai\u308ve"; // rendered as naïve
foreach (new TextIterator($text,
    TextIterator::CHARACTER) as $u) {
    var_inspect($u);
}
```

Result

```
unicode(1) "n" { 006e }
unicode(1) "a" { 0061 }
unicode(2) "ï" { 0069 0308 }
unicode(1) "v" { 0076 }
unicode(1) "e" { 0065 }
```

text iterator



Iterate over words in reverse order

```
$text = "Pouvez-vous me dire quelle heure il est ? Merci.";  
foreach (new ReverseTextIterator($text,  
                                TextIterator::WORD) as $u) {  
    if ($u != " ") echo($u), "\n";  
}
```

Result

```
.  
Merci  
?  
est  
il  
heure  
quelle  
dire  
me  
vous  
-  
Pouvez
```

text iterator



Truncate text at a word boundary

```
$it = new TextIterator($text, TextIterator::WORD);  
$offset = $it->preceding(40);  
echo substr($text, 0, $offset), "...\\n";
```

Get the last 2 sentences of the text

```
$it = new TextIterator($text, TextIterator::SENTENCE);  
$it->last();  
$offset = $it->previous(2);  
echo substr($text, $offset);
```

Get all the pieces delimited by boundaries

```
$it = new TextIterator($text, TextIterator::WORD);  
$words = $it->getAll();
```

text transforms



- Powerful and flexible way to process Unicode text
 - script-to-script conversions
 - normalization
 - case mappings and full-/halfwidth conversions
 - accent removal
 - and more
- Allows chained transforms

`[:Latin:] ; NFKD; Lower; Latin-Katakana;`

transliteration



```
$names = "  
    김, 국삼  
    김, 명희  
    たけだ, まさゆき  
    おおはら, まなぶ  
    Горбачев, Михаил  
    Козырев, Андрей  
    Καφετζόπουλος, Θεόφιλος  
    Θεοδωράτου, Ελένη  
";  
$r = strtotitle(str_transliterate($names, "Any", "Latin"));
```

```
Gim, Gugsam  
Gim, Myeonghyi  
Takeda, Masayuki  
Oohara, Manabu  
Gorbačev, Mihail  
Kozyrev, Andrej  
Kaphetzópoulos, Theóphilos  
Theodōrátou, Elénē
```

transliteration

- Here's how to get (a fairly reliable) Japanese pronunciation of your name

```
$k = str_transliterate('Britney Spears', 'Latin', 'Katakana');
echo($k),"\n";
$l = strtotitle(str_transliterate($k, 'Katakana', 'Latin')));
echo($l),"\n";
```

ブリテネイ スペアルス
Buritenei Supearusu



other things

- APC bundled
- PCRE default regex engine
- “taint” mode
- traits
- a couple of syntactic sugar treats
- 64-bit integer type
- general cleanup



pecl/intl

features



- Locales
- Collation
- Number and Currency Formatters
- Date and Time Formatters
- Time Zones
- Calendars
- Message Formatter
- Choice Formatter
- Resource Handler
- Normalization

versioning



- Works under PHP 5 and 6
- Uses native strings in PHP 6
- Requires UTF-8 strings in PHP 5
- Can use mbstring, iconv



Collator

comparing strings

● **compare(\$str1, \$str2) = -1, 0, 1**

```
$coll = new Collator("fr_CA");
if ($coll->compare("côte", "coté") < 0) {
    echo "less\n"; ←—————
} else {
    echo "greater\n";
}
```

côte < coté

sorting strings



- **sort(\$array, \$flags)**
- **asort(\$array, \$flags)**
- **sortWithSortKeys(\$array)**

```
$strings = array(  
    "cote", "côte", "Côte", "coté",  
    "Coté", "côté", "Côté", "coter");  
$coll = new Collator("fr_CA");  
$coll->sort($strings);
```

```
cote  
côte  
Côte  
coté  
Coté  
côté  
Côté  
coter
```

strength control

- **setStrength (\$strength)**

- **getStrength ()**

```
$coll = new Collator("fr_CA");
$coll->setStrength(Collator::PRIMARY);
if ($coll->compare("côte", "coté") == 0) {
    echo "same\n"; ←
} else {
    echo "different\n";
}
```

côte = coté



NumberFormatter



what it is

- allows formatting numbers as strings according to the localized format or given pattern or set of rules
- and parsing strings into numbers according to these patterns
- replacement for `number_format()`



formatter styles

123456.789 in en_US

- NumberFormatter::PATTERN_DECIMAL
123456.79 (with ##.##)
- NumberFormatter::DECIMAL
123456.789
- NumberFormatter::CURRENCY
\$123,456.79
- NumberFormatter::PERCENT
12,345,679%

• • • formatter styles

123456.789 in en_US

- NumberFormatter::SCIENTIFIC

1.23456789E5

- NumberFormatter::SPELLOUT

one hundred and twenty-three thousand, four hundred and
fifty-six point seven eight nine

- NumberFormatter::ORDINAL

123,457th

- NumberFormatter::DURATION

34:17:37

formatting

● **format(\$number [, \$type])**

```
$fmt = new NumberFormatter('en_US',
                           NumberFormatter::DECIMAL);
$fmt->format(1234);
// result is 1,234

$fmt = new NumberFormatter('de_CH',
                           NumberFormatter::DECIMAL);
$fmt->format(1234);
// result is 1'234
```



MessageFormatter



what it is

- produces concatenated messages in a language-neutral way
- operates on *patterns*, which contain *subformats*



what it is

- Need to get:

Today is November 21, 2007.

- Normal PHP way: `date('F d, Y')`
- MessageFormat would use
pattern: Today is {0,date}.
arg: `array(time())`

formatting

● **format(\$args)**

```
$pattern = "On {0,date} you have {1,number} meetings.";  
$args = array(time(), 2);  
$fmt = new MessageFormatter('en_US', $pattern);  
echo $fmt->format($args);  
  
// On November 22, 2007 you have 2 meetings.
```

formatting

● Trying different locales

```
$fr_pattern = "Aujourd'hui, {0,date,dd MMMM},  
                il y a {1,number} personnes sur {3}.";  
$fr_args = array(time(), 6579844000.0, 3, "la Terre");  
  
$msg = new MessageFormatter('fr_FR', $fr_pattern);  
echo $msg->format($fr_args);  
  
// Aujourd'hui, 22 novembre, il y a 6 579 844 000 personnes sur  
la Terre.
```



php 5.3

namespaces

- designed to solve scoping problems
- perhaps the most discussed issue ever on the internals list
- complex topic - still under work

namespaces



Definition

```
namespace Yahoo::News;  
class Dir {  
    ...  
}
```

Usage

```
// directly  
$foo = new Yahoo::News::Dir;
```

```
// import namespace  
use Yahoo::News;  
$foo = new News::Dir;
```

```
// rename class  
use Yahoo::News::Dir as YND;  
$foo = new YND;
```

```
// use global class  
$foo = new ::Dir;
```

lambdas and closures

- anonymous functions
- scope capture
- should be familiar to JS users

```
$lambda = function () {  
    echo "Hello World!\n";  
};  
$lambda();
```

lambdas and closures

more complex example

```
function getAdder($x) {  
    return function ($y) use ($x) {  
        return $x + $y;  
    };  
}  
  
$adder = getAdder(10);  
print_r( array_map( $adder, array(1, 2, 3) ) );
```

```
Array  
(  
    [0] => 11  
    [1] => 12  
    [2] => 13  
)
```

- PHp ARchive
- PharData = PDO for tar and zip files
- encapsulated applications

PharData



```
phar = new PharData('project.tar');

// add all files in the project
$phar->buildFromDirectory('./project');

// now compress it
$phar->convertToData(PHAR::TAR, PHAR::GZ);
```



access file in archive

```
include 'phar:///path/to/myphar.phar/file.php';
```

host an entire application

```
// create app archive
$phar = new Phar('myphar.phar');
$phar['cli.php']    = '<?php echo "Hello CLI World"; ?>';
$phar['index.php'] = '<?php echo "Hello Web World"; ?>';
$phar->setDefaultStub('cli.php', 'index.php');
```

extensions



- intl
- fileinfo
- sqlite3
- mysqlnd

- garbage collection
- NOWDOC
- limited GOTO
- ternary shortcut ?:
- numerous fixes, additions, and cleanups

Im in urendginn

playin wif ur stringz



obrigado

<http://gravitonic.com/talks>