

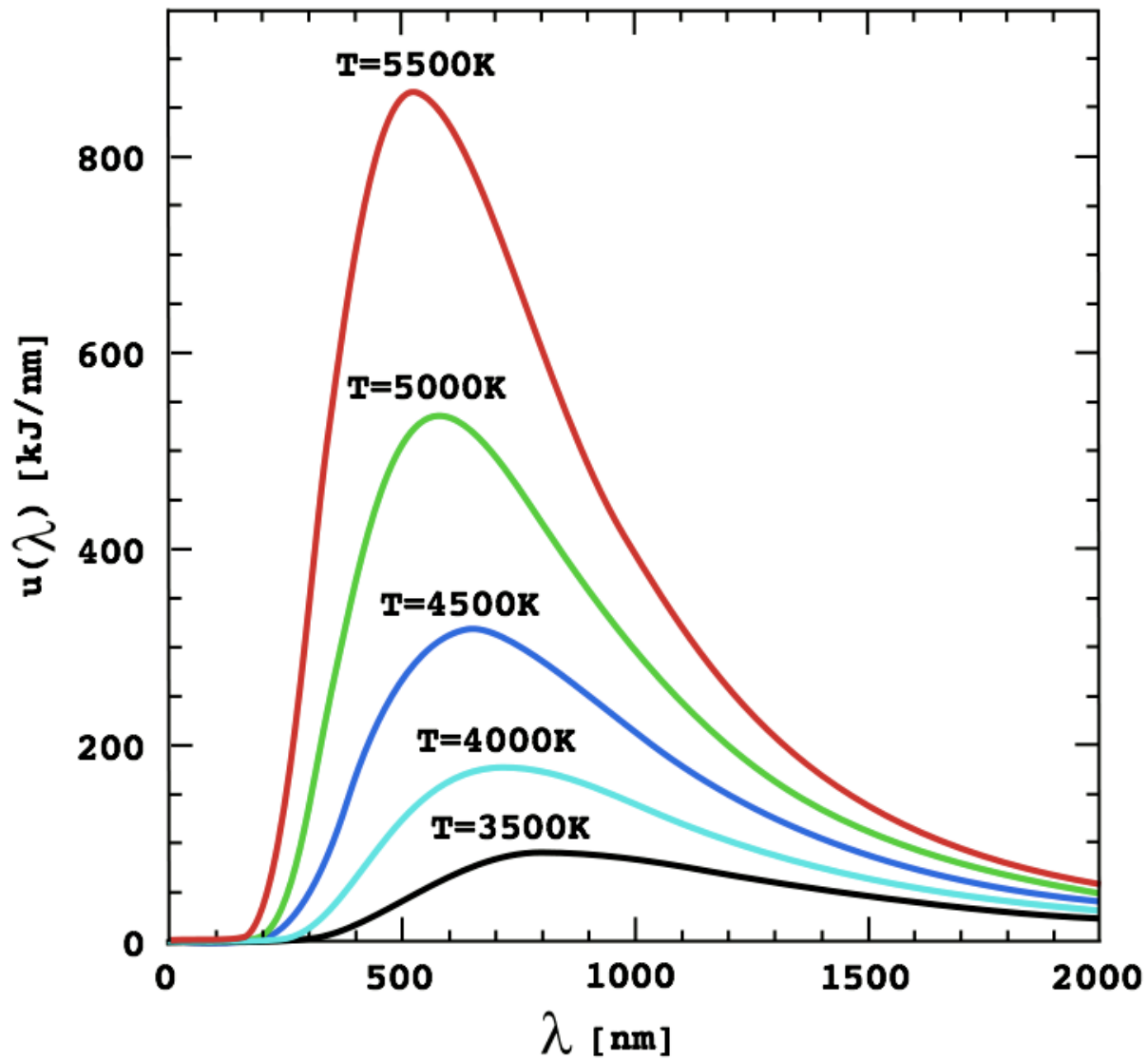


Andrei Zmievski
Chief Architect
Outspark, Inc

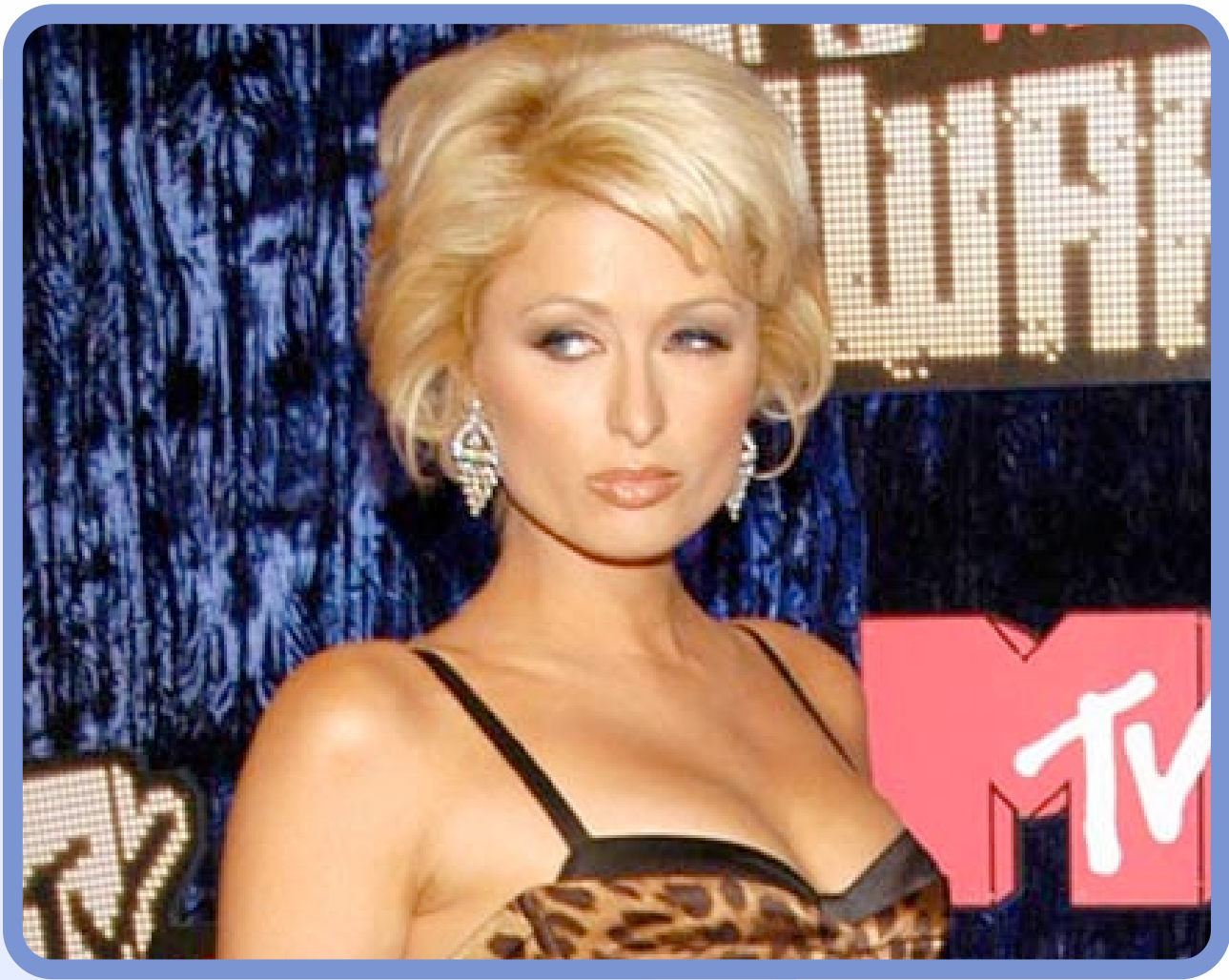
PHP ++

or, how to level up without the grind

GeekSessions, San Francisco



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





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?

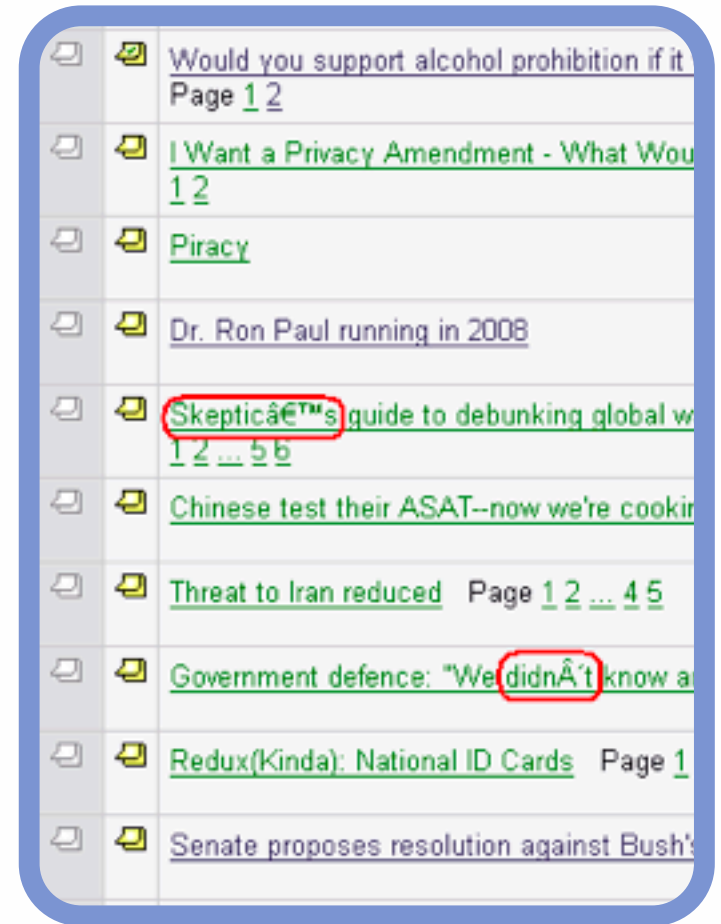


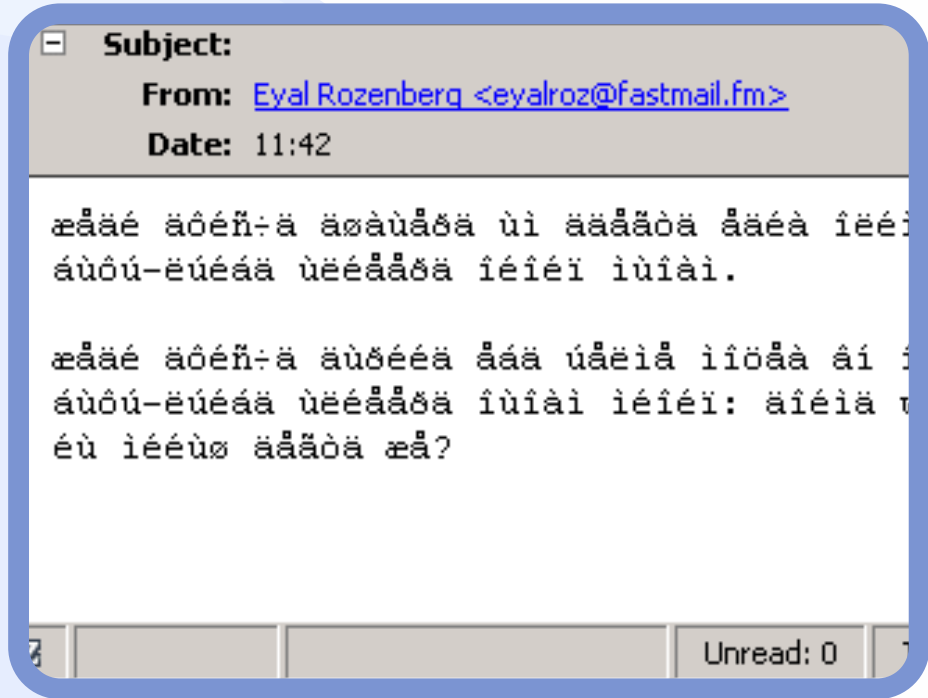
mojibake

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





mojibake

Encoding soup

1. provinciality
2. computer limitations
3. inertia

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

- 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 2007 (CLDR 1.5)
- 394 locales, with 135 languages and 149 territories



PHP 6

○ ● ● unicode support

● Everywhere:

- in the engine
- in the extensions
- in the API

○ ● ● unicode support

- Native and complete
 - no hacks
 - no mishmash of external libraries
 - no missing locales
 - no language bias

string types

- Unicode

- text

- default for literals, etc

- Binary

- bytes

- everything \notin Unicode type

string types ○ ● ●

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

- All string literals are Unicode

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

```
$jpn = "検索オプション"; // Unicode string  
echo strlen($jpn);      // 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
```

identifiers ○ ● ●

- Unicode identifiers are allowed

```
class コンポーネント {  
    function ັໂຮ ັໂຮ ັໂຮ { ... }  
    function சிவாஜி கணேசன் { ... }  
    function අසුරා ආඥා { ... }  
}  
  
$プロバイダ = array();  
$プロバイダ['הַשָּׂרָא הַגִּבּוֹר'] = 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 = "Việ\u0302\u0323t Nam"; // Việt Nam  
$str = strrev($u); // result is maN tệV,  
// not maN ệtV
```

streams ○ ● ●

- 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
```

- 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);
```



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`

```
$coll = new Collator("fr_FR");  
if ($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_FR");
$coll->sort($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

numeric collation ○ ● ●

● Collator::NUMERIC_COLLATION

```
$strings = array("10", "1", "2");  
$coll->setAttribute(Collator::NUMERIC_COLLATION,  
                  Collator::ON);  
$coll = new Collator(null);  
$coll->sort($strings);
```

1 < 2 < 10



NumberFormatter

○ ● ● what it is

- allows to format numbers as strings according to the localized format or given pattern or set of rules
- and parse strings into numbers according to the above 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



MessageFormatter

○ ● ● what it is

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

○ ● ● 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

● Can use various style modifiers

```
$pattern = "On {0,date,full} you received  
           {1,number,#,##0.00} emails.";  
$args = array(time(), 184);  
$fmt = new MessageFormatter('en_US', $pattern);  
echo $fmt->format($args);  
  
// On Tuesday, November 22, 2007 you received 184.00  
emails.
```

● ● ● 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.
```

○ ● ● other things

- APC bundled
- "taint" mode
- closures, aka "mind benders"
- a couple of syntactic sugar treats
- PCRE default regex engine
- 64-bit integer type
- general cleanup

Im in ur endginn

playin wif ur stringz





kthxbye

<http://gravitonic.com/talks>