THE GOOD, THE BAD, AND THE UGLY

What Happened to Unicode and PHP 6

Andrei Zmievski ❖ PHP Community Conference

ABOUT 1 YEAR AGO…

"Hello PHP 5.4, open for all new stuff." — Jani

TIME OF DEATH

March 11, 11:09:37 2010 GMT

5 YEARS EARLIER...

PHP 5.0.0
released in July 2004

5 YEARS EARLIER...

Firefox 1.0
released in November 2004

5 YEARS EARLIER...

Chrome
not even a twinkle in Google's eye

5 YEARS EARLIER...

Unicode
version 4.0.1

WHAT IS UNICODE?

and why do I need it?

Unicode

...is a computing industry standard for the consistent encoding, representation and handling of text expressed in most of the world's writing systems.

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 6.0 has 109,000+ characters
- Capacity for 1 million+ characters
- Widely supported by standards & industry

FEATURES

- Rich property set for every character
- Standard, unified encodings: UTF-8/16/32
- Extensive rules and documents for implementation
- Everything works, as long as everyone follows the rules

UNICODE != I18N

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

TIME FORMATS

- USA: `4:00 P.M.`
- France: `16.00`
- Japan: `1600`
- Don’t forget to identify the time zone

CURRENCY

- Symbol placement
- Symbol length (1-15)
- Number width
- Number precision:
  - Spain, Japan – 0
  - Mexico, Brazil – 2
  - Egypt, Iraq – 3

US $12.34

12.345,67 €

12$34€

¥123

SORTING

- Swedish: z < ö
- German: ö < z
- Dictionary: öf < of
- Phonebook: of < öf
- Upper-first: A < a
- Lower-First: a < A
- Contractions: H < Z, but CH > CZ
- Expansions: OE < Œ < OF

CLDR

- Hosted by Unicode Consortium
- Latest release: December 2010 (CLDR 1.9)
- 516 locales, with 187 languages and 166 territories

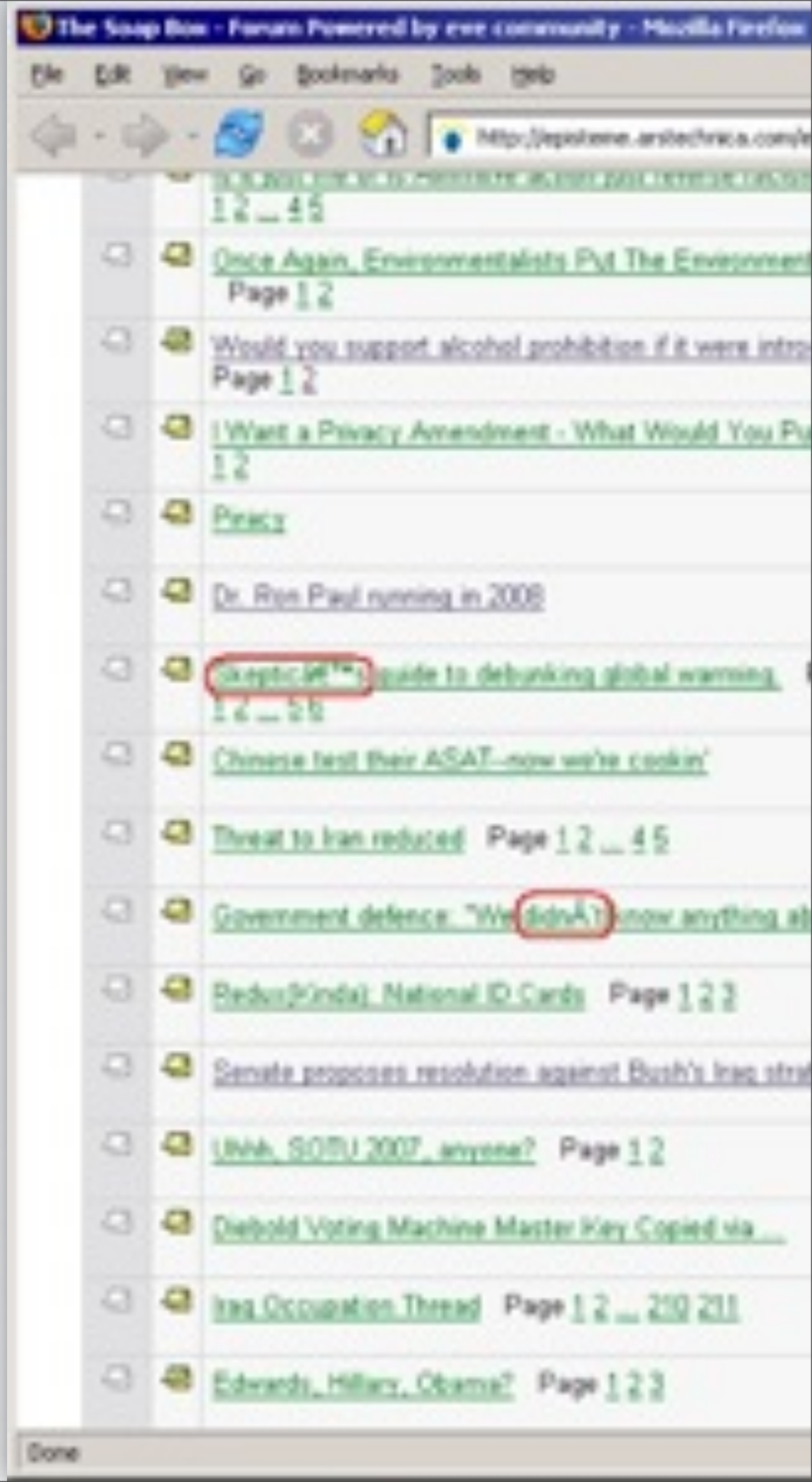
WHY WEB NEEDS UNICODE

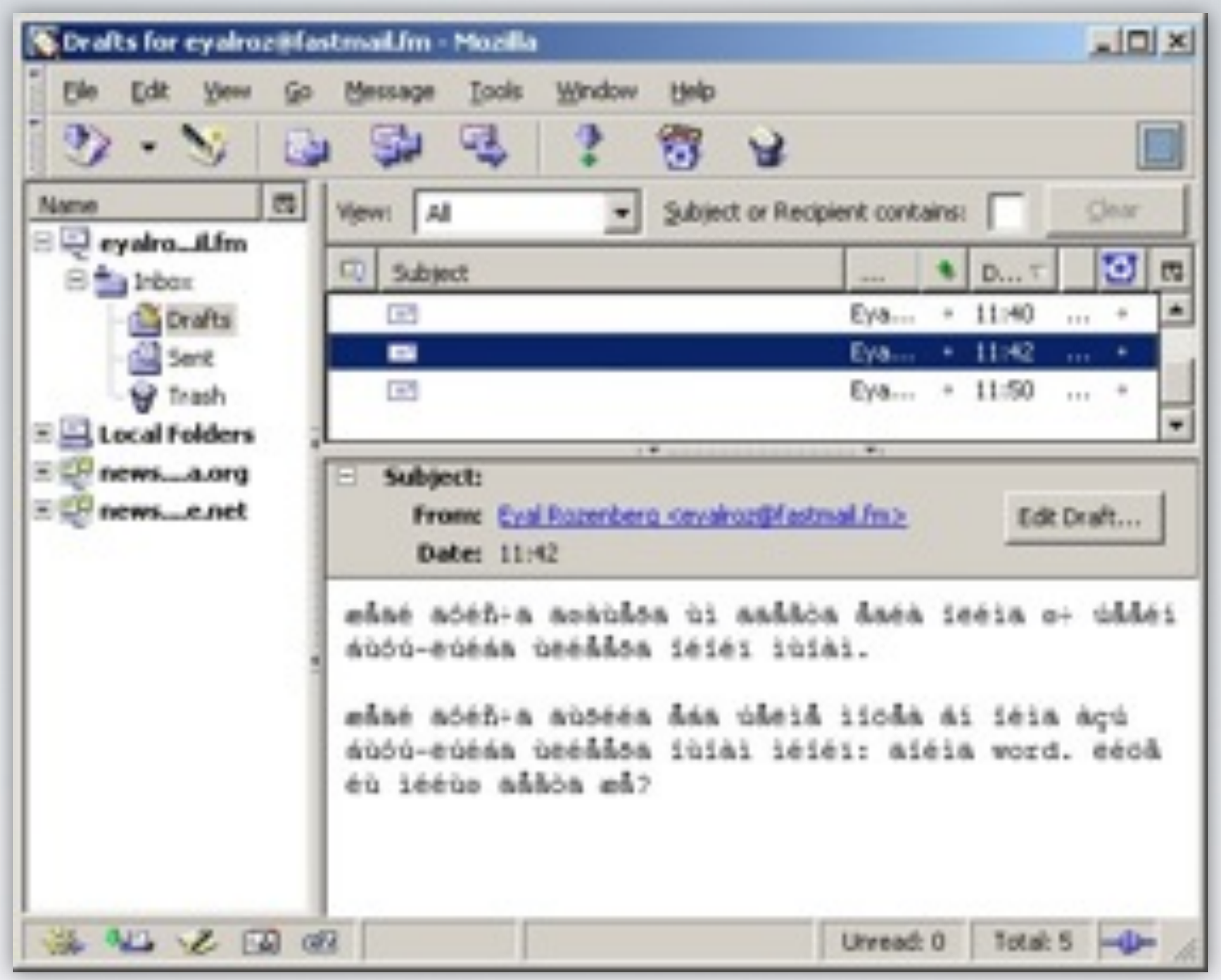
MOJIBAKE

もじばけ

MOJIBAKE

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




Drafts for eyalroz@fastmail.fm - Mozilla
File Edit View Go Message Tools Window Help
Name
eyalroz@fastmail.fm
Inbox
Drafts
Sent
Trash
Local Folders
news...a.org
news...e.net
View: All
Subject or Recipient contains:
Clear
Subject
D...
Eya... 11:40
Eya... 11:42
Eya... 11:50
Subject:
From: Eyal Rozenberg <eyalroz@fastmail.fm>
Date: 11:42
Edit Draft...
Unread: 0
Total: 5

MOJIBAKE

MOJIBAKE

I ♥ UNICODE,

YOU ♥ UNICODE

I | UNICODE,
YOU | UNICODE

Helgi

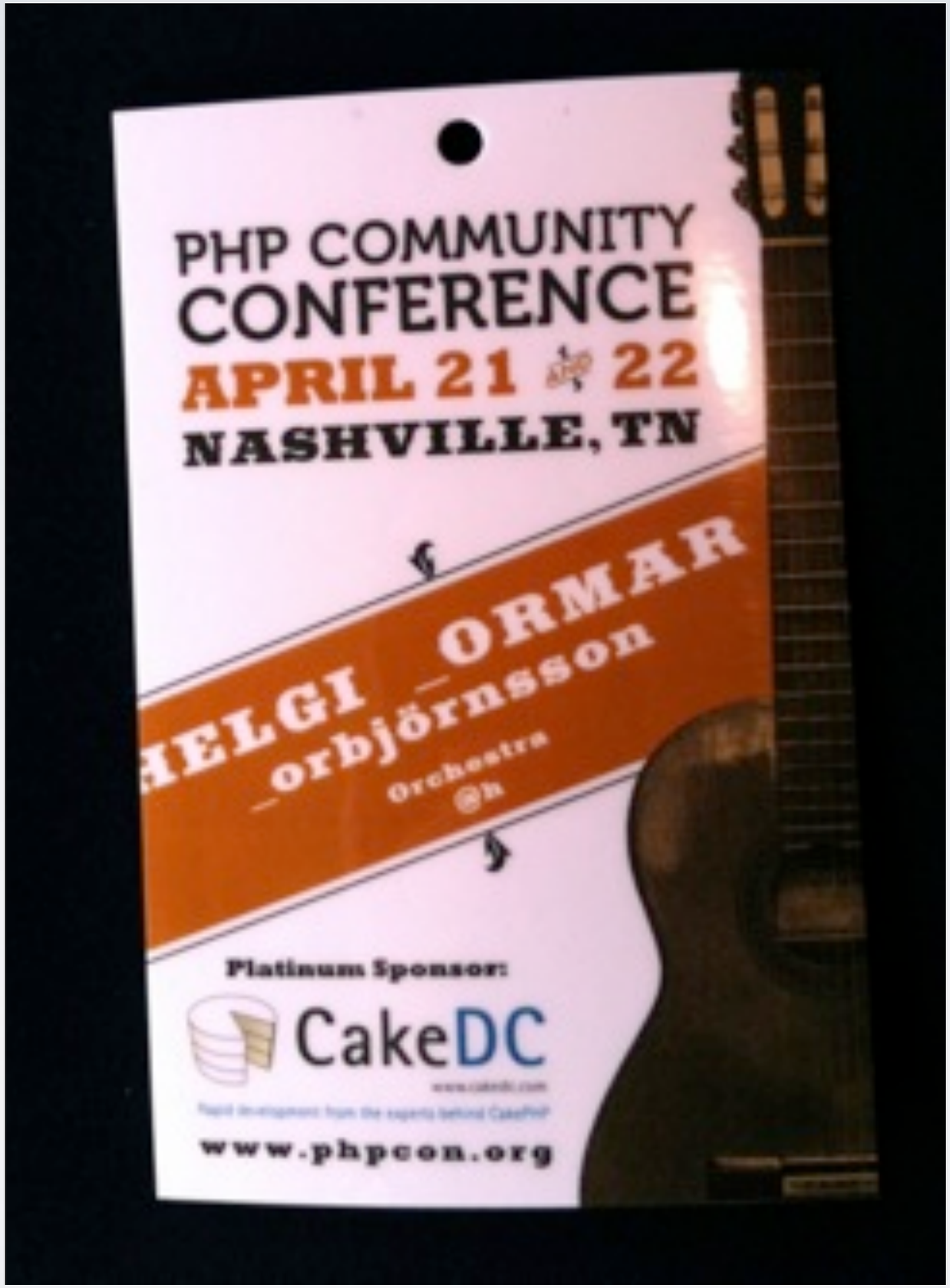
Helgi

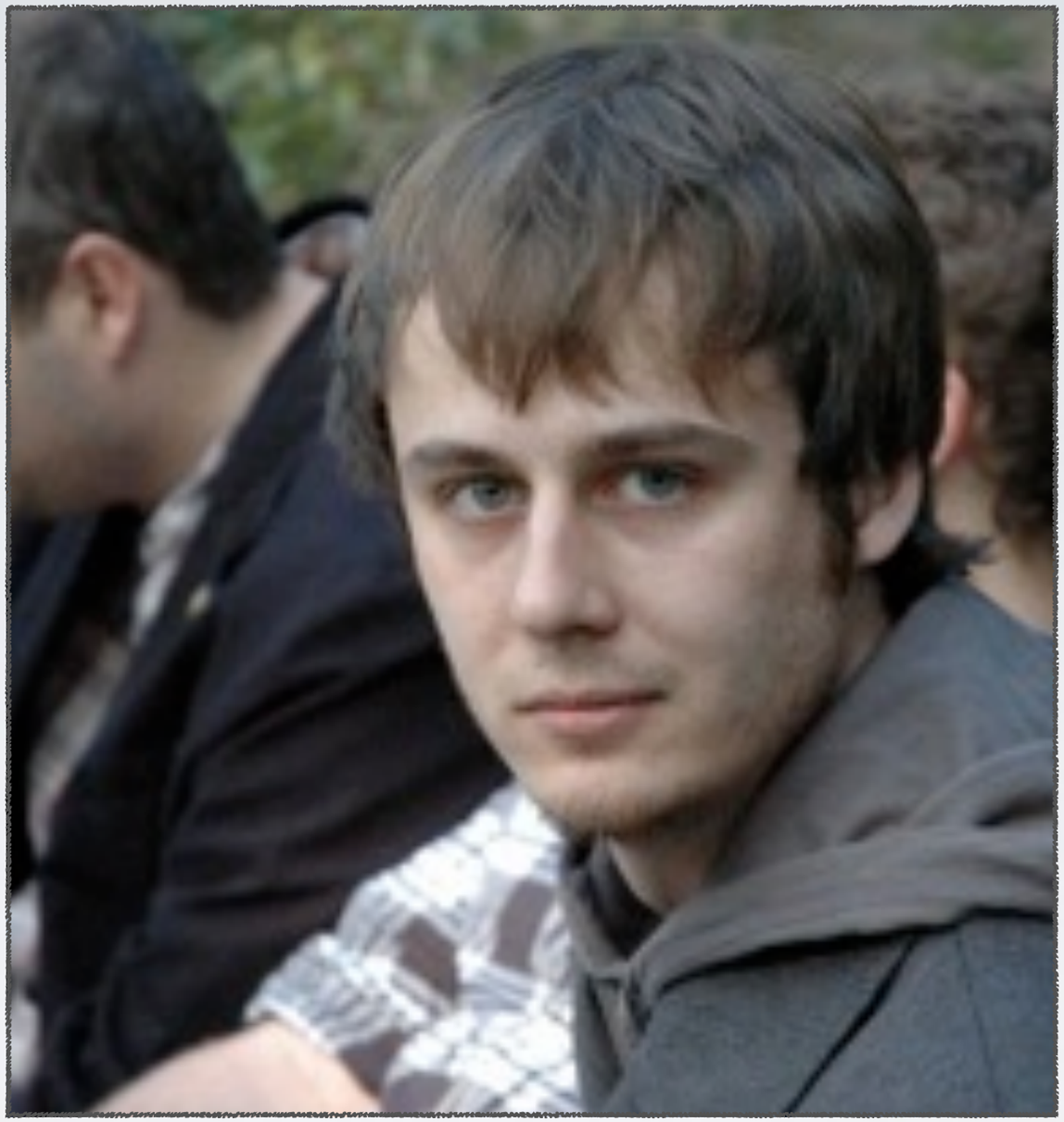
Helgi Þormar Þorbjörnsson

ISLTHORP

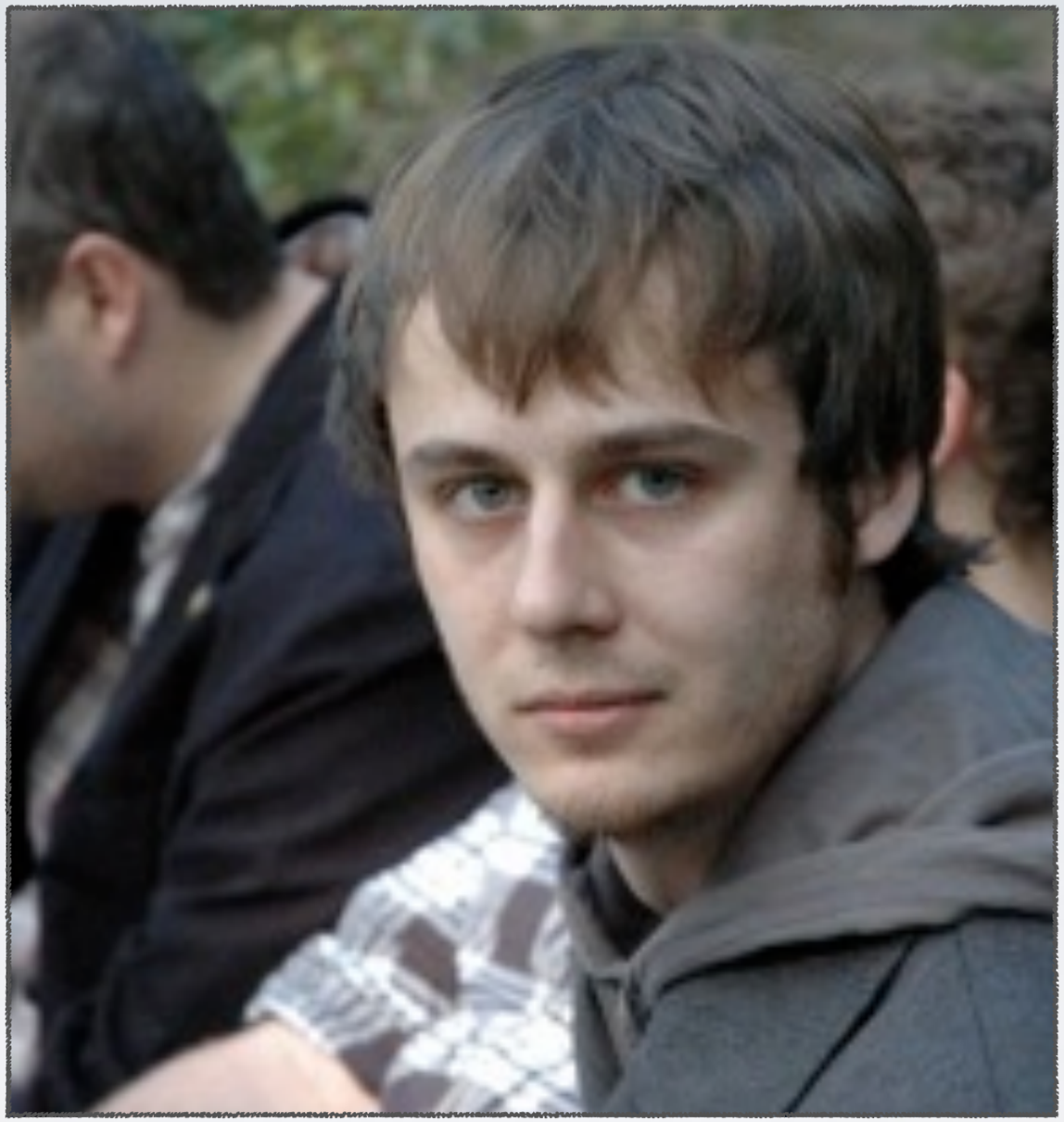
OR

MR. SECURITY OVERRIDE

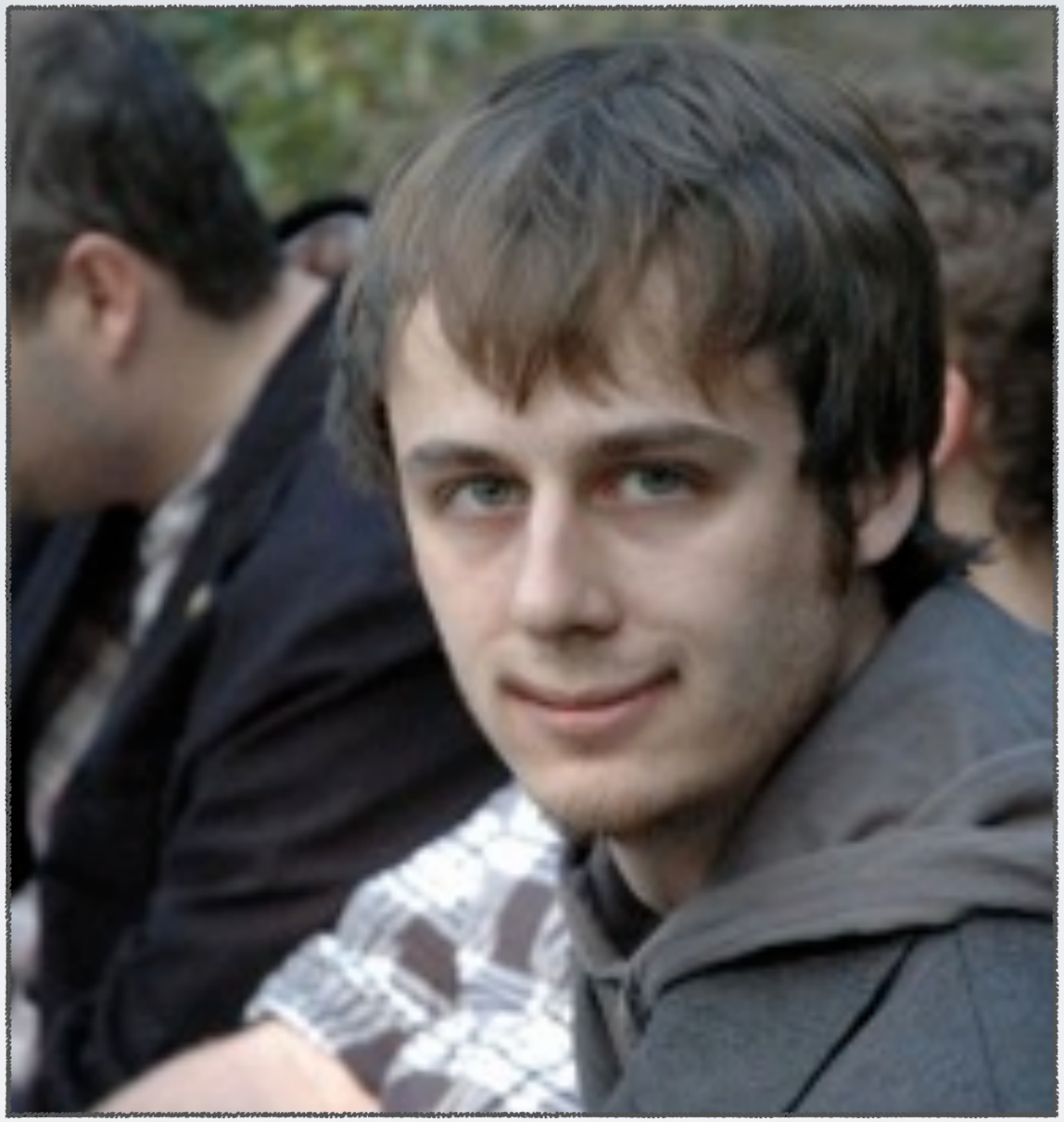

PHP COMMUNITY
CONFERENCE
APRIL 21 to 22
NASHVILLE, TN
HELGI _ORMAR
_orbjörnsson
Orchestra
@h
Platinum Sponsor:
CakeDC
www.phpcon.org



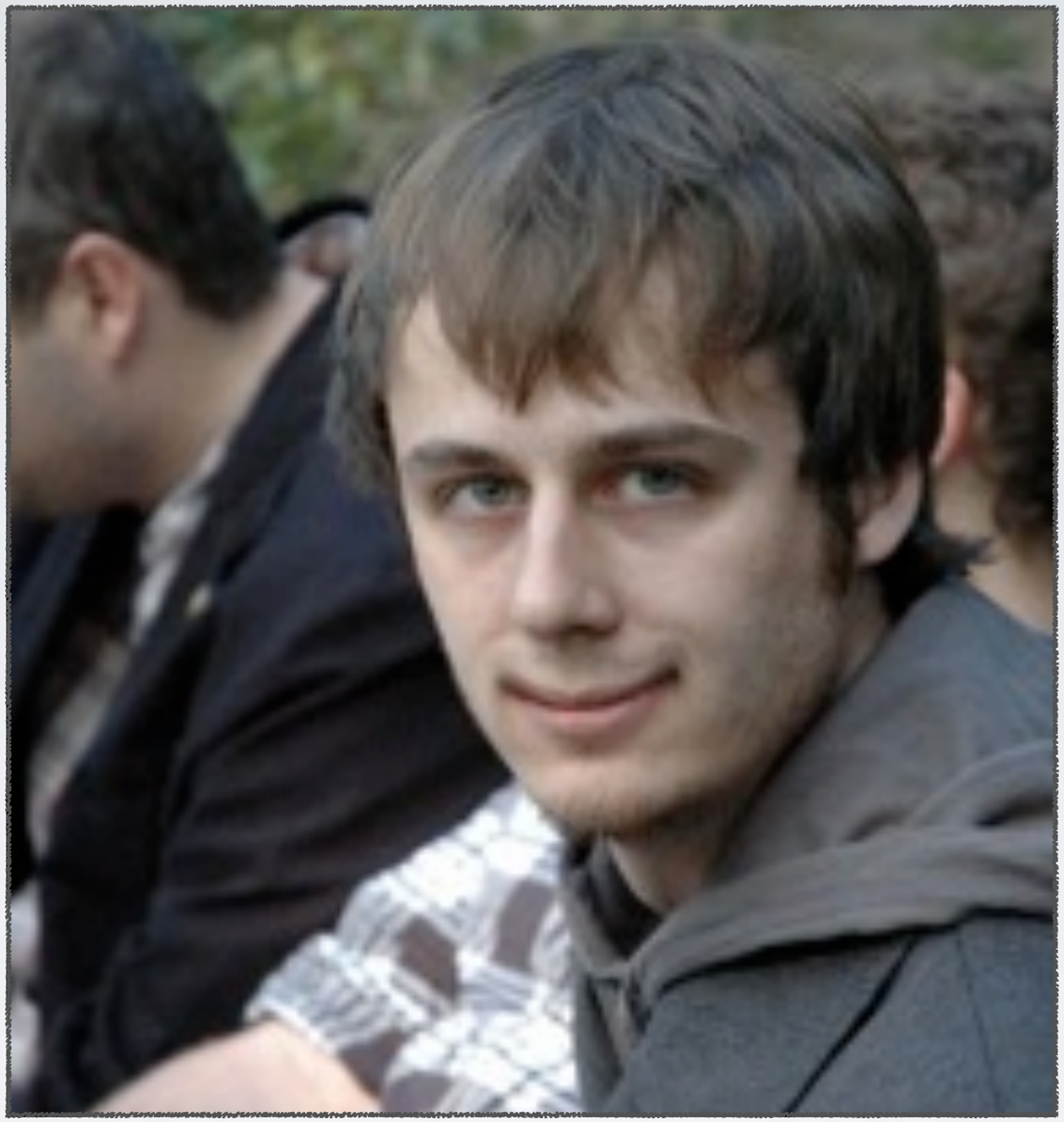
Joel



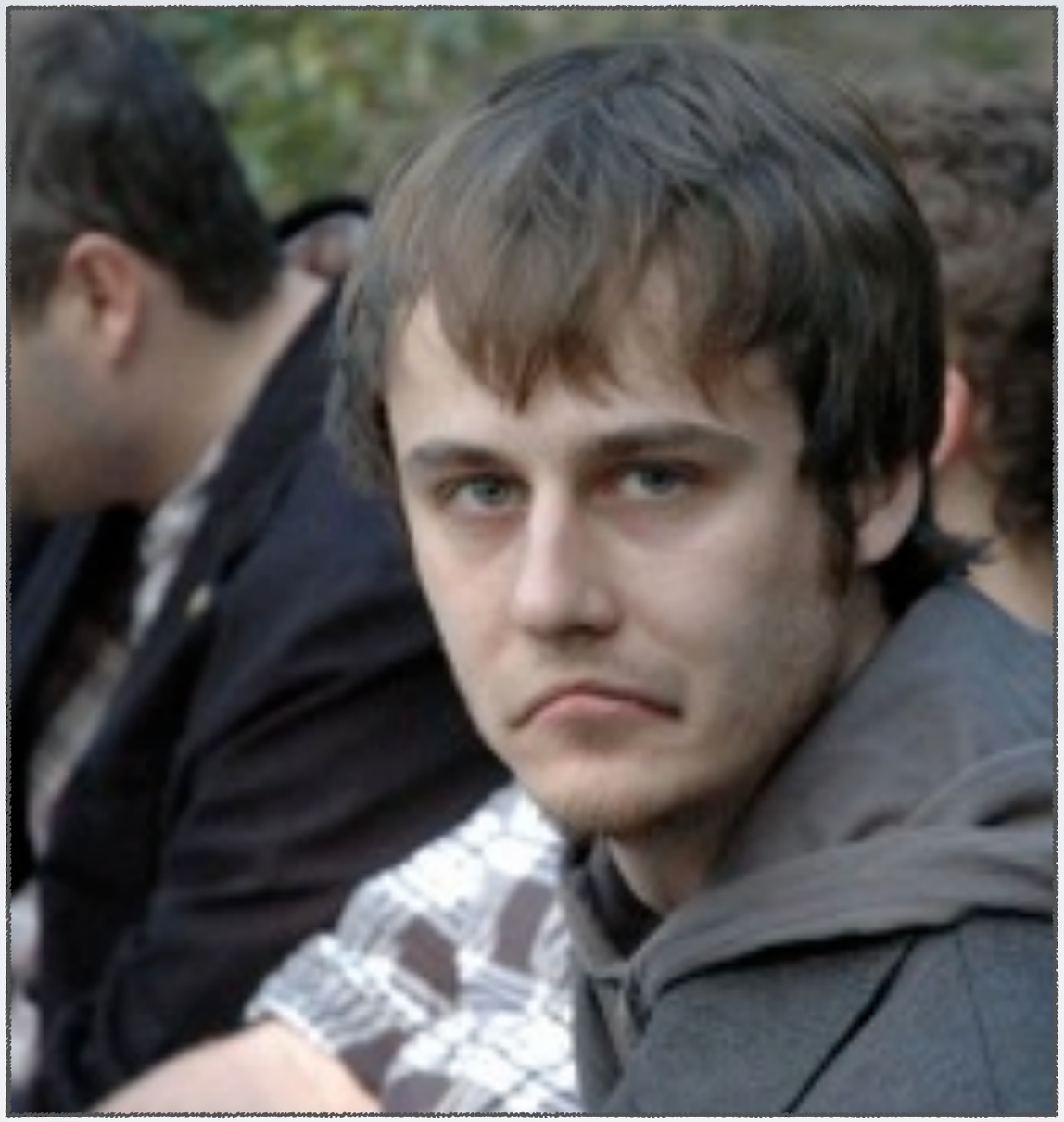
Joël



Joël



JoÃ«l



JoÃ«l

WEAKEST LINK

PRINCIPLE APPLIES

WHY PHP NEEDS UNICODE

PHP

- Essential Web platform
- Since Web needs Unicode…
- …so does PHP
- Do not want to be the weakest link

THE PROJECT

THE PROJECT

- Launched in February 2005 by me at Yahoo
- Small group from Yahoo, Zend, and PHP development community
- Design before code

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

ICU LIBRARY

International Components for Unicode

- ✓ Unicode Character Properties
- ✓ Unicode String Class & text processing
- ✓ Text transformations (normalization, upper/lowercase, etc)
- ✓ Text Boundary Analysis (Character/Word/Sentence Break Iterators)
- ✓ Encoding Conversions for 500+ legacy encodings
- ✓ Language-sensitive collation (sorting) and searching
- ✓ Unicode regular expressions
- ✓ Thread-safe
- ✓ Formatting: Date/Time/ Numbers/Currency
- ✓ Cultural Calendars & Time Zones
- ✓ (230+) Locale handling
- ✓ Resource Bundles
- ✓ Transliterations (50+ script pairs)
- ✓ Complex Text Layout for Arabic, Hebrew, Indic & Thai
- ✓ International Domain Names and Web addresses
- ✓ Java model for locale-hierarchical resource bundles. Multiple locales can be used at a time

THE PROJECT

- Development was in a separate repository
- Merged into PHP tree once the basics were working
- Initially slated for 5.x
- Extensive changes necessitated a major version bump

PHP 6 = PHP 5 + Unicode

PHP 5 = PHP 6 - Unicode

Unicode = PHP 6 - PHP 5

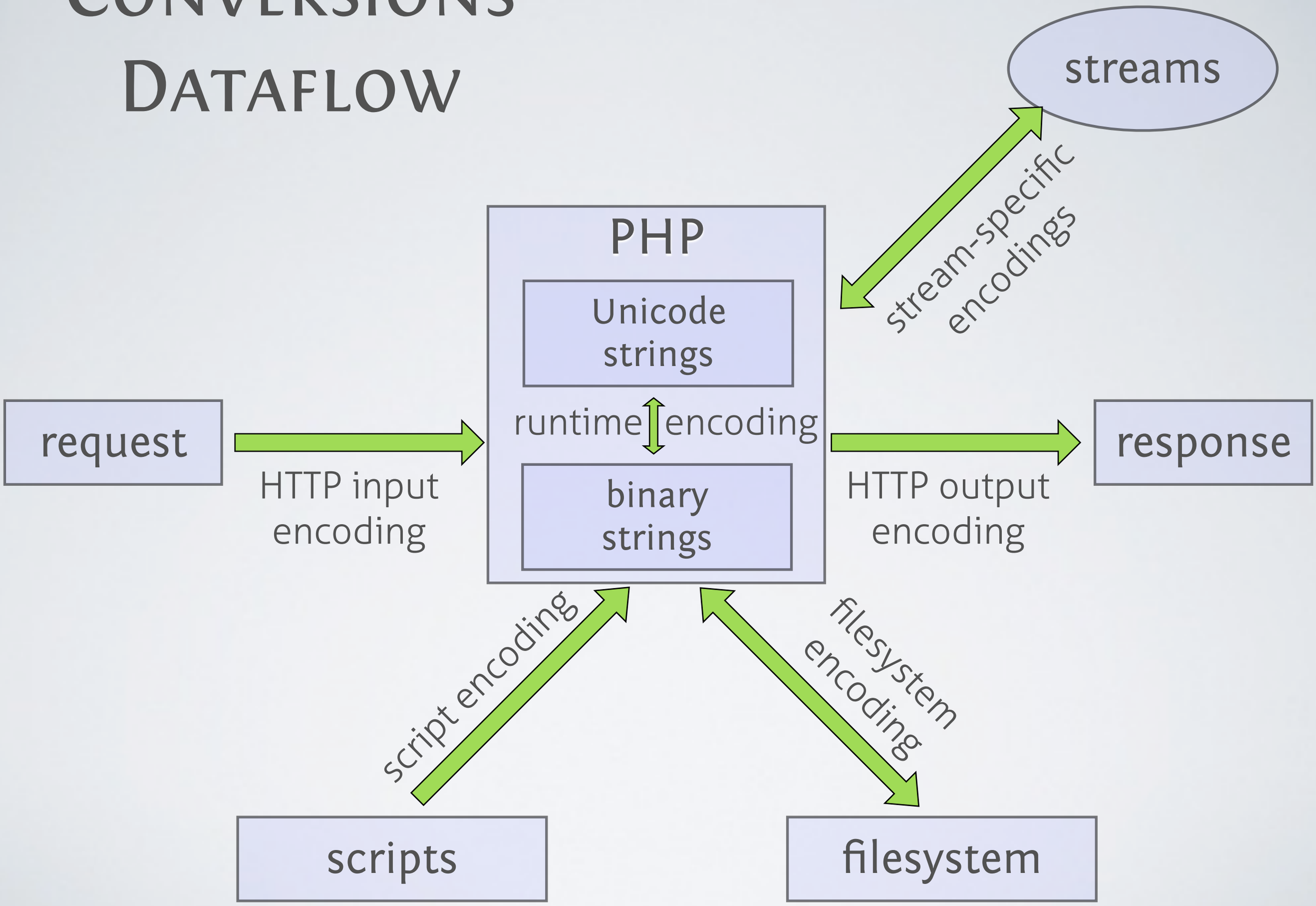
PHP 6

PHP 6

STRING TYPES

- Unicode
  - text
  - default for literals, etc
- Binary
  - bytes
  - everything ∉ Unicode type

Conversions Dataflow



STRINGS

- String literals are Unicode
- 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 ᓱᓴᐧ ᐊᓪᓚᖁᖅᓴᖅ()    { ... }
    function சிவாஜி கணேசன்()  { ... }
    function འཇིག་ཡུལ།()          { ... }
}

$プロバイダ = array();

$プロバイダ['רַעְיוֹנוֹת שָׁנָה'] = new コンポーネント;
```

FUNCTIONS

- Functions understand Unicode text and apply appropriate rules
- i.e. case manipulation

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

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

TRANSLITERATION

```php
$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
```

PECL/INTL

FEATURES

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

COLLATION

sorting

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

result

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

NUMBER FORMATTING

123456.789 in en_US

- NumberFormatter::DECIMAL
  123456.789

- NumberFormatter::CURRENCY
  $123,456.79

- NumberFormatter::ORDINAL
  123,457th

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

MESSAGE FORMATTING

with modifiers

```
$pattern = “On {0,date,full} you received
                {1,number,#,##0.00} emails.”;
$args = array(time(), 1184);
$fmt = new MessageFormatter(‘en_US’, $pattern);
echo $fmt->format($args);
```

result

```
On Tuesday, November 22, 2007 you received
1,184.00 emails.
```

POSTMORTEM

WHAT WENT RIGHT

1. RAISED AWARENESS

- Spoke at multiple conferences about the project
  - including Unicode Conference
- Shoved Unicode down people's throats at every opportunity

2. CHOSE THE RIGHT TECH

- ICU library had everything we needed
- Low- and high-level functionality
- Good support from its developers

3. UNIT TESTS

- Every function handling strings had to be ported
- Unit tests showed us where things broke
- Also easy to track progress

4. PECL/INTL EXTENSION

- A lot of i18n/l10n functionality in a self-contained extension
- Ensuring that it worked with PHP 5

5. CODE SEGREGATION

- Proof-of-concept developed by only a few people
- Faster decisions, iteration, development
- Things slowed down after merging into the main tree
  - but was necessary to spread the workload

WHAT WENT *RONG*

1. CHOICE OF UTF-16

Thought to be the best compromise

UTF-8

- Backward-compatible with ASCII
- Avoids complications of endianness
- Dominant UTF encoding for the Web
- Supported in a lot of libraries, APIs, etc

UTF-8, BUT...

- Variable-length encoding (1-4 bytes)
- Uses 3 bytes for BMP code points > U+07FF
- Not all byte sequences are valid
- ICU did not have many UTF-8 APIs (at the time)
  - on-the-fly conversion is necessary

UTF-32

- Uses exactly 4 bytes for each code point
  - directly indexable!

UTF-32, BUT...

- Uses exactly 4 bytes for each code point
  - 4x the size of UTF-8 for majority of languages
- Only affordable by people from rich oil countries
- Still needs conversion to UTF-16 when using ICU
- Endianness

UTF-16

- “65,536 code points should be enough for everyone…”
- 2 bytes to represent all of BMP (U+0 to U+FFFF)
  - directly indexable in that plane
- Internal encoding of ICU

UTF-16, BUT…

- Requires *surrogate pairs* for code points › U+FFFF
  - still variable-length
- 2x the size of UTF-8 for Latin, Greek, Cyrillic, Armenian, Hebrew, Arabic and other scripts
- Can’t be manipulated by normal C string handling
- Endianness

CHOICE OF UTF-16

- Thought that CJK languages would benefit from UTF-16
- Primary driver was the ICU APIs
- Problems: no direct indexing, many conversions
- Would probably choose UTF-8, if started over
  - no need for decoding/encoding on the periphery
  - can be used by C-based libraries

2. CRUCIAL CODE LAGGED

2. CRUCIAL CODE LAGGED

- PDO (and native DB extensions)
- filter
- proper substring search (collation-based)
- some `ext/standard` functionality

3. LACK OF MINDSHARE

3. LACK OF MINDSHARE

- Probably <10 people who understood the intricacies of the Unicode and ICU
- In the end, implementation deemed too technically difficult
- People were bored converting large chunks of already working code

4. DELAYED NEW FEATURES

5. MEA CULPA

END GAME

RE-ORG

- PHP 6 trunk was moved to a branch
- PHP 5.4 became the trunk
- Kick-started development of new features
- Some clean-ups and improvements from 6 back-ported to 5.4

PEOPLE MATTER

- The project ran out of steam
  - PHP development culture means that people work on what they're interested in
  - Clearly, the Unicode/i18n implementation wasn't interesting enough to be viable

INTERNALS

"Because it's nearly impossible to participate on Internals if your poo-throwing arm isn't strong."

— @coates

PERSISTENCE

"Those with talent, competence, energy, and good ideas over a period of time tend to be the main drivers behind PHP development."

— me

PERSISTENCE

"Those with talent, competence, energy, and good ideas over a period of time

and who outlast the rest

tend to be the main drivers behind PHP development."

— me

PROGRESS

- No development on the Unicode branch
- No visible effort to develop alternatives

FUTURE?

- Lighter, gentler implementations?
  - mbstring is clunky
  - separate Unicode String class would also be clunky
- Open field for someone with a great idea, persistence, and people skills

STEPPING AWAY

- Invalidation of several man-years of hard work is discouraging
- Did not feel like pushing the project up the hill again
- Working on more fun stuff these days

LESSONS LEARNED

- Rewriting large existing code base is hard
- Making people do tedious stuff is hard
  - make it interesting for them (game-like)
- Waiting for results of long iterations is hard
  - short, results-oriented projects (if possible)
- Stay committed

FINITA LA COMEDIA

http://joind.in/3349 ❖ http://zazzle.com/andreiz