PHP Norge - Oslo, Norway

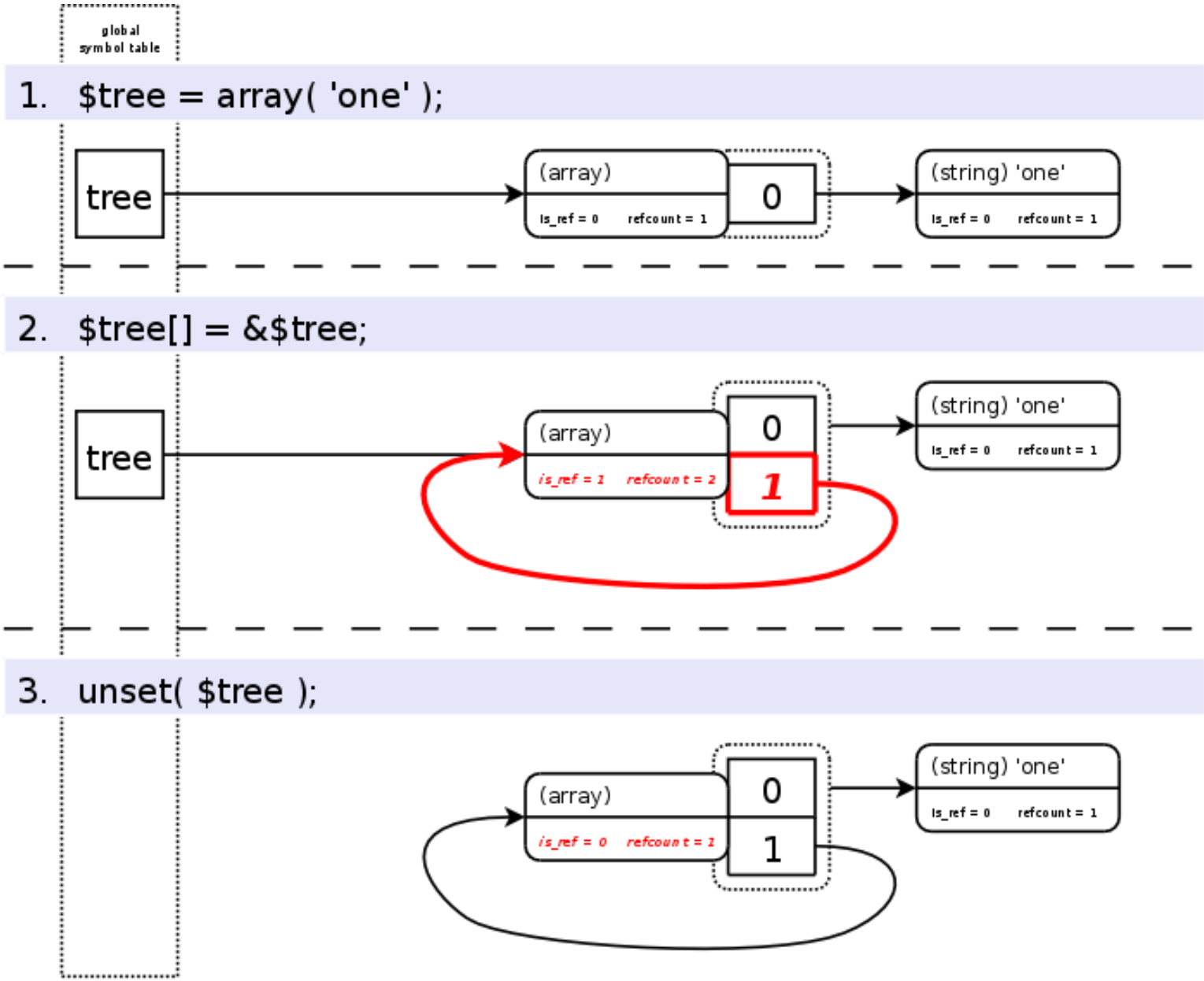
Derick Rethans - dr@ez.no

http://derickrethans.nl/talks.php

The non-Unicode things

(Actually, also in PHP 5.3)

Variables
Circular References



Garbage Collection

- Collects all the variable containers into a buffer
- Once buffer is full, the algorithm runs to reclaim space
- Can be turned on and off at will: gc_enable(), gc_disable()
- Drastically reduces memory usage for testing eZ Components (2.4GB->500MB)

```
GC Statistics
-------------
Runs:              124
Collected:          2434961
Root buffer length: 4
Root buffer peak: 10000

    Possible            Remove from Marked
        Root    Buffered     buffer     grey
    -------- -------- ----------- ------
ZVAL 13233645 2444651    2106657 18967079
ZOBJ 41671687 2120597    1218587 2475517
```

Namespaces

- Solves name clashes
- Provides name aliases and resolving
- Can contain constants, classes and functions

```
<?php
namespace eZ;

const DEVELOPMENT = 42;

class Configuration
{
}

function bootstrap()
{
}
?>
```

Namespaces
Using namespaces: use

- the use operator makes a namespaces available

```
<?php
use eZ::Database::Connection as DbConnection;

use eZ::Database;

new DbConnection(); // uses eZ::Database::Connection
new Database::Driver(); // would use eZ::Database::Connection
?>
```

Namespaces
Name resolution rules

All qualified names (A::B::C) are translated during compile-time.

```
<?php
use eZ::Database::Driver;

// translated at compile-time to eZ::Database::Driver::Sqlite()
new Driver::Sqlite();
?>
```

All unqualified names (C) are translated during compile-time.

```
<?php
use eZ::Database::Driver;

// translated at compile-time to eZ::Database::Driver()
new Driver();
?>
```

Namespaces
Name resolution rules

Inside a namespace, all unqualified names (C) are translated during run-time.

```
<?php
namespace eZ::Database;

// translated at run-time first to eZ::Database::Driver()
new Driver();
?>
```

If not found, it looks for the internal class Driver.
If not found, it attempts to autoload eZ::Database::Driver.
In order to use the global class Driver, use "new ::Driver()."

Namespaces
Quirks

You can not override internal classes:

ez_datetime.php
```
<?php
namespace eZ;

class DateTime
{
}
?>
```

test.php
```
<?php
require 'ez_datetime.php';

use eZ::DateTime;
?>
```

Displays:

```
Fatal error: Cannot use eZ::DateTime as DateTime because the
name is already in use in /tmp/test.php on line 4
```

The Unicode things

Unicode?

Ç'është Unicode?, – ዩኒኮድ ምንድን ነው? – – ؟ "يونِكود" ما هي الشفرة الموحدة
ইউনিকোড কী? – ዩኒኮድ ውረኒ ግን? – Какво е Unicode ? – 什么是Unicode(S — S)? –
Što je Unicode? – Co je Unicode? – Hvad er Unicode? – Wat is Unicode? –
Kio estas Unikodo? – Mikä on Unicode? – Qu'est ce qu'Unicode? –
რა არის უნიკოდი? – Was ist Unicode? – Τι είναι το Unicode; –
Tí eἶναι tὸ Unicode; – מה זה יוניקוד (Unicode)? – यूनिकोड क्या है? –
Hvað er Unicode? – Que es Unicode? – Cos'è Unicode? –
오니코드에 대해 ? – Kas tai yra Unikodas? – Што е Unicode? –
X'inhu l-Unicode? – Czym jest Unikod? – O que é Unicode? –
Ce este Unicode? – Что такое Unicode? – ዩኒኮድ ምቃሩ? – Шта је Unicode? –
Kaj je Unicode? – ¿Qué es Unicode? – Vad är Unicode? –
யூனிக்கோடு என்றால் என்ன?, – యూనీకోడ్ అంటే ఏమిటి?, – Unicode คืออะไร? –
ዩኒኮድ እንታይ ኢዩ? – Što je Unicode? – Evrensel Kod Nedir? –
– ؟همه نن دیگه یونِکود Unicode dégen néme? – Unicode là gì? –
Beth yw Unicode? – ዩኒኮድ ወረኻ የኻ?

i18n challenges
InternationalizatioN

- Support for multiple encodings: conversion, detection, processing...
- Support for multiple language in different encodings and scripts

Sorting Strings
The French Are Weird

How would you sort: côté (side), côte (coast), cote (dimension), coté (with dimensions)?
Logical is: cote, coté, côte, côté
But the french do it like: cote, côte, coté, côté

The french are not the only ones with "weird" sorting!

- In Lithuanian, y is sorted between i and k.
- In traditional Spanish ch is treated as a single letter, and sorted between c and d.
- In Swedish v and w are considered variant forms of the same letter.
- In German dictionaries, öf would come before of. In phone books the situation is the exact opposite.

Do We Need Something New?

- PHP only deals with bytes, not characters.
- PHP doesn't know anything about encodings.
- Having a binary image in a string is nice, but not if you need to deal with i18n

Iconv

Character Set Conversions

- Is part of glibc
- BSD requires an external library
- Enabled by default in PHP 5
- Supports 100s of character sets

But it doesn't solve:
localization, sorting, searching, encoding detection and you still work with binary data only!

mbstring
Multi Byte Support!

- Handles certain encoding problems for you
- Updates string functions by "overloading" them

But it doesn't solve:
localization, sorting, searching and you still work with binary data only!

Anything Else That Sucks?

Locales in PHP

- Some of PHP's functions can make use of POSIX-locales
- But that's only a few of them
- Those locales are system dependent (different names, rules, etc...)
- They are not always available

Unicode and ISO-10646 (UCS)

- UCS uses 31 bits for character storage
- Contains all known characters and symbols
- First 128 characters are the same as ASCII
- First 256 characters are the same as ISO-8859-1
- Unicode 3.0 describes the BMP (Basic Multilingual Plane) (16 bits)
- Unicode 3.1 describes other planes (21 bits)
- Characters are ordered in language/script blocks: Basic latin, Cyrillic, Hebrew, Arabic, Gujarati, Runic, CJK etc.
- Encoding in numerous encodings: UCS-2, UCS-4, UTF-8, UTF-16 etc.

A Њ ش א 𐍆н 媛

Building New Characters

You can compose new characters from base characters with combining modifiers that use no "space".

Equivalents:

```
å != å
U+00C5 != U+0041 + U+030A
```

Alternative Order:

```
a +ˆ  + ̣ = ậ
a + ̣ +ˆ  = ậ
```

Unicode Is More

- Unicode is a multi-linguage character set
- Standard encodings: UTF-8, UTF-16 and UTF-32
- Defines algorithms for plenty of issues (Collation, Bidi, Normalization)
- It defines properties for characters:

```
Å
00C5;LATIN CAPITAL LETTER A WITH RING ABOVE;Lu;0;L;0041 030A;;;;
    N;LATIN CAPITAL LETTER A RING;;;00E5;

DŽ
01C4;LATIN CAPITAL LETTER DZ WITH CARON;Lu;0;L;<compat> 0044 017D;;;;
    N;LATIN CAPITAL LETTER D Z HACEK;;;01C6;01C5

Å
212B;ANGSTROM SIGN;Lu;0;L;00C5;;;;
    N;ANGSTROM UNIT;;;00E5;
```

What Do We Want for PHP?

- Native Unicode strings
- A clear separation between Binary / Native (Encoded) Strings and Unicode Strings
- Unicode string literals
- Updated language semantics
- Where possible, upgrade the existing functions
- Backwards compability
- PHP should do what most people will expect
- Make complex things possible, without making using strings in PHP complex
- Must be as good as Java's support

How is It Going To Work?

- UTF-16 as internal encoding
- All functions and operators work on Normalized Composed Characters (NFC)
- All identifiers can contain Unicode characters
- Internationalization is explicit, not implicit
- You can turn off Unicode semantics if you don't need it

UTF 16 Surrogates

As UTF-16 is supposed to encode the full Unicode character set.

- UTF-16 uses a double two-byte sequence for characters outside the BMP
- Special ranges in the Unicode range are used for this

```
byte 1 = 0xd800 – 0xdbff
byte 2 = 0xdc00 – 0xdfff
```

```
U+1D31D TETRAGRAM FOR JOY
        0xD834 0xDF1D
```

- Code point: a character
- Code unit: a two-byte sequence with UTF-16

How Will It Be Implemented?

ICU: International Components for Unicode

- Unicode is extremely complex, with ICU we don't have to implement it ourselves
- ICU has a lot of features, is fast, stable, portable, extensible, Open Source and well maintained and supported

ICU Features

- Character, String and Text processing
- Text Transformations
- Encoding Conversions
- Collation
- Localization: date, time, number, currency formatting

Roadmap

- Support Unicode in the Engine
- Upgrade existing functions: 60% done
- Add new functions for explicit i18n/l10n support
- Expose ICU's features

http://php.net/~scoates/unicode/render_func_data.php

eZ
Share your information

How Do We Turn It On?

- With an INI setting: unicode_semantics
- It can not be turned on "per-vhost"
- (Almost) no behavioral changes when it's not enabled
- The setting does not mean you won't have any Unicode strings

String Types

- string: Used to represent binary data, for example the contents of a JPEG file or a "native" string.

```
. P N G . . . . . . . . I H D R
89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52
B l å     b æ     r ø     l
62 6C C3 A5 62 C3 A6 72 C3 A8 6C
```

- unicode: Strings, internally encoded in UTF-16.

```
B     l     å     b     æ     r     ø     l
62 00 6C 00 E5 00 62 00 E6 00 72 00 F8 00 6C 00
```

String Literals

string -> string, unicode

Unicode Semantics are off:

```
<?php // script is encoded in UTF-8
$str = "hallo daar!";
echo gettype($str), ': ', strlen($str), "\n";

$str = "привет!";
echo gettype($str), ': ', strlen($str), "\n";
?>
```

outputs:

```
string: 11
string: 13
```

Unicode Semantics are on:

```
<?php // script is encoded in UTF-8
$str = "hallo daar!";
echo gettype($str), ': ', strlen($str), "\n";

$str = "привет!";
echo gettype($str), ': ', strlen($str), "\n";
?>
```

outputs:

```
unicode: 11
unicode: 7
```

Characters, not Bytes!

- All functions and operators work on Code Points (characters) and not Code Units (bytes)
- Backward compatible if you only used single byte encodings before
- This does create overhead though, as we need to scan through a whole string

String Indexes:

```
<?php

    $string = "网C 搜索G;
    // bytes are: E7 BD 91 E9 A1 B5 E6 90 9C E7 B4 A2

    echo $string[1];
?>
```

ICU Locales

- ICU comes with it's own Locale information
- PHP currently uses POSIX locales for some functions only
- Those functions need to be modified

ICU Locales
Example

```
<?php
    locale_set_default("nl");
    echo strtotitle("het ijsselmeer (ijsselmeer) is ßaf"), "<br/>\n";

    locale_set_default("tr");
    echo strtotitle("het ijsselmeer (ijsselmeer) is ßaf"), "\n";
?>
```

ICU Locales
Example

Comparing strings:

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

Ignore case and accents:

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

ICU Locales

Array Sorting Example

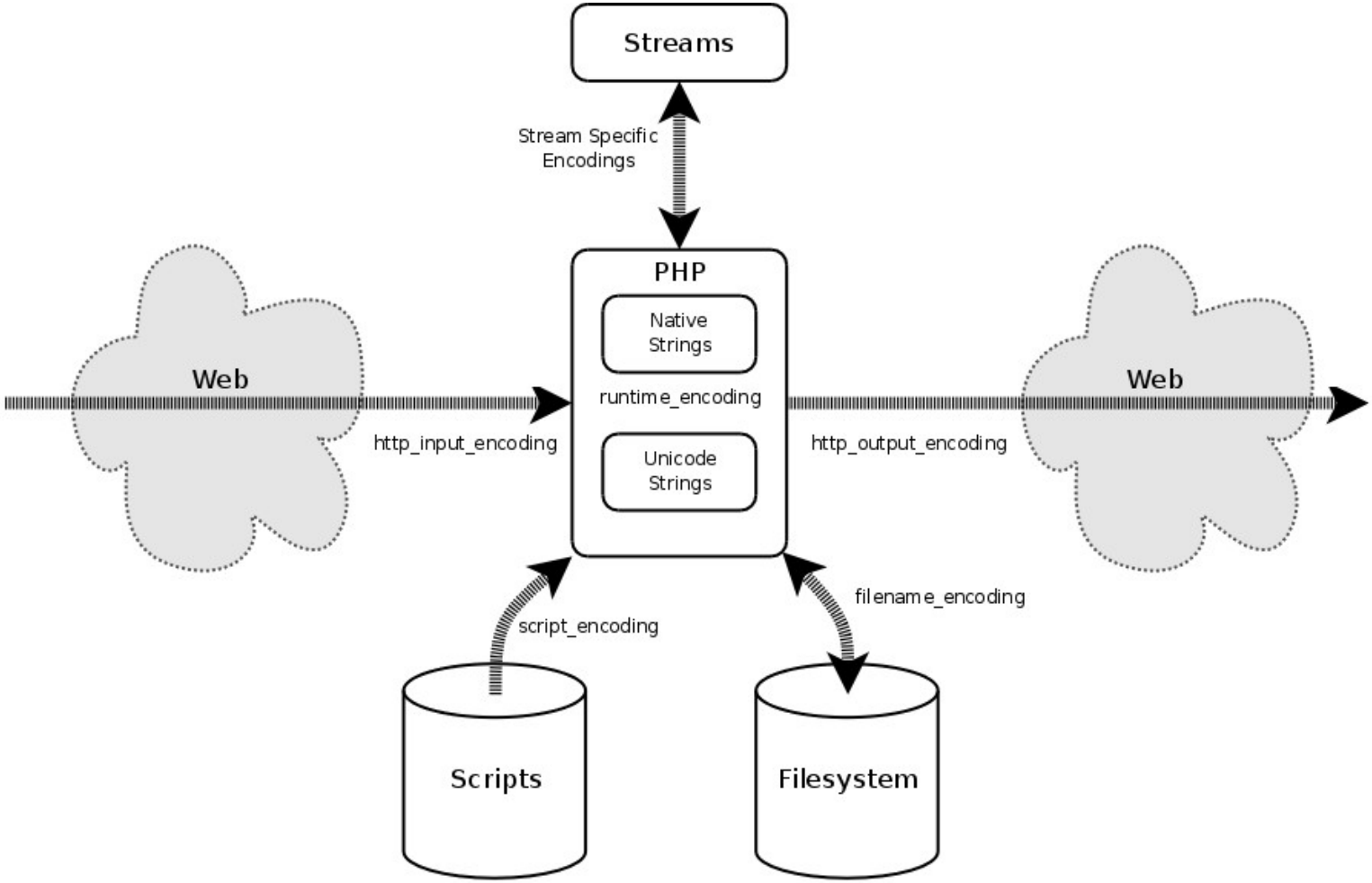
```
<pre># orig norm loc trad
-------------------------
<?php
    $d = $c = $b = $a = array('mapa', 'kilo', 'libro', 'llave', 'loca');
    sort($b);
    locale_set_default('es_VE');
    sort($c, SORT_LOCALE_STRING);
    locale_set_default('es_VE@collation=traditional');
    sort($d, SORT_LOCALE_STRING);

    for ($i = 0; $i < 5; ++$i) {
        echo sprintf('%d. %-5s %-5s %-5s %-5s<br/>',
            $i + 1, $a[$i], $b[$i], $c[$i], $d[$i]);
    }
?>
```

Text Iteration
Example

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

Encodings Overview



Script Encoding

- Is used by the parser to read in your script
- Determines how string literals and identifiers are handled
- Can be set with an ini setting (script_encoding) or with an inline "pragma"
- No matter what the script's encoding is, the resulting string is always a Unicode string (or identifier)

Script Encoding

Setting Script Encoding With declare()

Interpreting an iso-8859-1 script as UTF-8:

```
<?php
    declare(encoding="iso-8859-1");

    $str = "blå = 青 ";
    var_inspect($str);
?>
```

Interpreting an UTF-8 script as UTF-8:

```
<?php
    declare(encoding="utf-8");

    $str = "blå = 青 ";
    var_inspect($str);
?>
```

Script Encoding

```
<?php
class Person {
    public $firstName;
    public $lastName;
    public $birthday;

    function calculateAge() {
        $age = (time() - $this->birthday) / (365.2422 * 86400);
        return floor($age);
    }
}

$me = new Person;
$me->birthday = strtotime( "1978-12-22 09:15" );
echo "I am " . $me->calculateAge() . " years old.";
?>
```

Script Encoding

```
<?php
class 人 {
    public $名;
    public $姓;
    public $誕生日;

    function 年齢を計算() {
        $年齢 = (time() - $this->誕生日) / (365.2422 * 86400);
        return floor($年齢);
    }
}

$私 = new 人;
$私->誕生日 = strtotime( "1978-12-22 09:15" );
echo "私は " . $私->年齢を計算() . " 才です。 ";
?>
```

Runtime Encoding

- Determines which encoding to attach to Native Strings
- Also used when functions are not upgraded to support Unicode yet

Encoding problems:

```
<?php
    ini_set('unicode.runtime_encoding', 'iso-8859-5');
    ini_set('unicode.from_error_subst_char', '2D');
    $food = 'blåbær != блабар';
    $food2 = (binary) $food;
    $food3 = (string) $food2;
    echo $food, '<br/>', $food2, '<br/>', $food3, "<br/>\n";
?>
```

HTTP Output Encoding

- Is used as encoding for the output of the script
- Script output is encoding on the fly
- Binary strings will never be automatically converted

On the fly encoding:

```
<?php
    declare(encoding="iso-8859-1");
    ini_set('unicode.output_encoding', 'utf-8');

    $str = "rødbærsyltetøy<br/>";
    $bstr = b "rødbærsyltetøy<br/>";

    echo $str, $bstr;
?>
```

Output:

rÃ¸dbÃ¦rsyltetÃ¸y
rødbærsyltetøy

Streams I/O

- PHP's IO uses a unified layer: streams
- There is no clue in many cases what encoding a file is in
- By default, PHP opens streams in binary mode, and no encoding conversion is done

Text Streams

t is no longer only for Windows line endings.
Default encoding is used, which is by default UTF-8. The following reads 42 UTF-8 characters and returns them as a Unicode string:

```
<?php
$f = fopen( "somefile.txt", "rt" );
$str = fread( $f, 42 );
?>
```

Text Streams
Default context

You can change the default encoding like:

```
<?php
stream_default_encoding( "iso-8859-1" );
$str = file_get_contents( "somefile.txt", FILE_TEXT );
?>
```

Or use a specific context for a stream:

```
<?php
$ctxt = stream_context_create(
    NULL, array( 'encoding' => 'koi8-r' )
);
file_put_contents( "someotherfile.txt", $data, FILE_TEXT, $ctxt );
?>
```

Or set the encoding after opening a stream:

```
<?php
$f = fopen( "someotherfile.txt", 'r' );
stream_encoding( $f, 'iso-8859-5' );
?>
```

Transliteration

```
<?php

$names = "でりっく
        デリック
        Дерик Ретханс";

echo nl2br( strtotitle( str_transliterate( $names, 'Any', 'Latin' ) ) );
?>
```

```
<?php
$t = str_transliterate( 'Derick Rethans', 'Latin', 'Katakana' );
$b = str_transliterate( $t, 'Katakana', 'Latin' );

echo $t, ' ', $b;
?>
```

Fallback Encoding

- Used if any of the other encoding settings is not set
- Easy way of configuring all encoding settings
- Defaults to UTF-8 if it's not set

INI Settings Recap:

```
unicode.script_encoding = "UTF-8"
    Source encoding for your script

unicode.runtime_encoding = "iso-8859-15"
    Internal encoding used for "native strings"

unicode.from_error_subst_char = "2f"
    Hex value of substitution character

unicode.http_input_encoding = "UTF-8"
    Default encoding for HTTP input variables

unicode.output_encoding = "UTF-8"
    Encoding used for script output

unicode.fallback_encoding = "UTF-8"
    Fallback encoding
```

When Can We Have This?

- When it is ready.
- Development version is in CVS, get snapshots.
- Release: preview release at the end of 2006.

eZ
Share your information

Resources

- PDM notes: http://php.net/~derick/meeting-notes.html
- This presentation: http://derickrethans.nl/talks.php
- Questions: dr@ez.no