Strings usage: so many tools are already in your hands!

SymfonyCon Brussels 2023 - Marion Hurteau
@marionherisson / mhurteau@jolicode.com

Hello World 👋

Marion Hurteau

@MarionHerisson

/MarionLeHerisson

mhurteau@jolicode.com

Hello World

👩‍💻 JoliCode since 2019

➡️ Consulting, production, audit, expertise and training

🏰 Poney club, castle & home-made beer

Drinking alcohol is dangerous for your health. Drink in moderation (and in good company)

What is a string?

“What is a string?” is a string.

💁 💬 Anything writable, printable, or earable really

💻 💬 0 and 1

👩‍💻 ❓

Glyph

~ is a glyph and so are n or ñ

→ an image, an abstract form

Grapheme

~ is a grapheme and n is another grapheme

→ a minimally distinctive unit of writing

→linked to the context of a particular writing system

Grapheme cluster

ñ is a grapheme cluster

→ think of it as a character

Grapheme

ñ is a grapheme cluster

- a minimally distinctive unit of writing in the context of a particular writing system

- think of it as a character

A bit of history 🔎

The first encodings

💻 💬 « 01100001 » means « a » 💁 💬

🙅 Or does it?

The first encodings

1963 : ANSI : ASCII ! 7 bits → 127 characters

ASCII

Bits b_4	b_3	b_2	b_1	Row ↓ / Column →	0 (000)	1 (001)	2 (010)	3 (011)	4 (100)	5 (101)	6 (110)	7 (111)
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	|
1	1	0	1	13	CR	GS	—	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	_	o	DEL

Not cool for the rest of the world

German : Schildkr**ö**te 🐢

Swedish : Sk**å**l! 🍻

French : **É**l**é**phant 🐘

The first encodings

1963 : ANSI : ASCII ! 7 bits → 127 characters

1972 : 8 bits CPUs are here ! 8 bits → 255 characters

Common character encodings [edit]

- ISO 646
  - ASCII
- EBCDIC
- ISO 8859:
  - ISO 8859-1 Western Europe
  - ISO 8859-2 Western and Central Europe
  - ISO 8859-3 Western Europe and South European (Turkish, Maltese plus Esperanto)
  - ISO 8859-4 Western Europe and Baltic countries (Lithuania, Estonia, Latvia and Lapp)
  - ISO 8859-5 Cyrillic alphabet
  - ISO 8859-6 Arabic
  - ISO 8859-7 Greek
  - ISO 8859-8 Hebrew
  - ISO 8859-9 Western Europe with amended Turkish character set
  - ISO 8859-10 Western Europe with rationalised character set for Nordic languages, including complete Icelandic set
  - ISO 8859-11 Thai
  - ISO 8859-13 Baltic languages plus Polish
  - ISO 8859-14 Celtic languages (Irish Gaelic, Scottish, Welsh)
  - ISO 8859-15 Added the Euro sign and other rationalisations to ISO 8859-1
  - ISO 8859-16 Central, Eastern and Southern European languages (Albanian, Bosnian, Croatian, Hungarian, Polish, Romanian, Serbian and Slovenian, but also French, German, Italian and Irish Gaelic)
- CP437, CP720, CP737, CP850, CP852, CP855, CP857, CP858, CP860, CP861, CP862, CP863, CP865, CP866, CP869, CP872
- MS-Windows character sets:
  - Windows-1250 for Central European languages that use Latin script, (Polish, Czech, Slovak, Hungarian, Slovene, Serbian, Croatian, Bosnian, Romanian and Albanian)
  - Windows-1251 for Cyrillic alphabets
  - Windows-1252 for Western languages
  - Windows-1253 for Greek
  - Windows-1254 for Turkish
  - Windows-1255 for Hebrew
  - Windows-1256 for Arabic
  - Windows-1257 for Baltic languages
  - Windows-1258 for Vietnamese
- Mac OS Roman
- KOI8-R, KOI8-U, KOI7
- MIK
- ISCII
- TSCII
- VISCII
- JIS X 0208 is a widely deployed standard for Japanese character encoding that has several encoding forms.
  - Shift JIS (Microsoft Code page 932 is a dialect of Shift_JIS)
  - EUC-JP
  - ISO-2022-JP
- JIS X 0213 is an extended version of JIS X 0208.
  - Shift_JIS-2004
  - EUC-JIS-2004
  - ISO-2022-JP-2004
- Chinese Guobiao
  - GB 2312
  - GBK (Microsoft Code page 936)
  - GB 18030
- Taiwan Big5 (a more famous variant is Microsoft Code page 950)
  - Hong Kong HKSCS
- Korean
  - KS X 1001 is a Korean double-byte character encoding standard
  - EUC-KR
  - ISO-2022-KR
- Unicode (and subsets thereof, such as the 16-bit 'Basic Multilingual Plane')
  - UTF-8
  - UTF-16
  - UTF-32
- ANSEL or ISO/IEC 6937

The

"Internet"

Mojibake

ãƒã‚°ã‚¤ã‚ —ã ¦ã „ã ¾ã ›ã‚“ ãƒˆãƒ¼ã‚¯ æŠ•ç¨¿è¨˜éŒ² ã‚¢ã‚«ã‚¦ãƒ³ãƒˆä½œæˆ ãƒã‚°ã‚¤ã‚³ã‚¸ãƒ¥ãƒˆãƒšãƒｰｼ ãƒŽãƒｰﾄ é–²è¦§ ç·¨é›†† å±¥æ´è¡¡ç¤º Wikipediaå†…ã‚‘æ¤œç´¢

ウィキペディア
フリー百科事典

ãƒ¡ã‚¤ãƒšãƒｰｼ
ã‚³ãƒŸｭﾆﾃｨ･ﾎﾟｰﾀﾙ
æœ€è¿‘ ®å‡ºæ ¥ｶﾞ
æ–°ã —ã „ãƒšãƒｰｼ
æœ€è¿‘ ®æ›´æ–°
ã Šã ¾ã ‹ã ›è¡¨ｼ
ç·´ç¿’ｼﾞﾍﾞﾝﾁﾏｰｸ
ã‚¢ｯﾌﾟﾛｰﾄﾞ
(ã‚¦ã‚£ｷﾒﾃﾞｨｱ･ｺﾓﾝｽ)

ãƒˆｰﾙ

ãƒˆｰﾙ
ｶﾃｺﾘｰ
ã ŠçŸ¥ã‚‰ã ›
ãƒ ã‚°ã ®å ±ｶﾞ
ﾗﾝﾀﾞﾑ
ã‚¦ｨｷﾒﾃﾞｨｱ･ｺﾓﾝｽﾄｱ ã „å ˆã‚ã ›

ãƒ„ｰﾙ

ãƒﾍﾙﾌﾟ･ﾍﾙﾌﾟ
ã ™ã‚‹ﾘﾝｸ
ｶﾚﾝﾀﾞｰ
ｱｯﾌﾟﾛｰﾄﾞﾌｧｲﾙ
ﾌﾟﾘﾝﾄ･ｴｸｽﾎﾟｰﾄ

ãƒ¡ﾆｭｰｰ

ãƒˆｰｸ
ﾌｧｲﾙ
ﾘﾝｸﾁｪｯｸ
ﾌｫｰﾏｯﾄ
ﾍﾙﾌﾟ

æ–‡å—åŒ–ã ‘

å‡ºå…¸: ãƒ•ãƒªｰ™¾ç§‘ä°‹å…¸ã€Žã‚¦ã‚£ｷﾒﾃﾞｨｱ¼‰ã€

W ã‚¦ã‚£ｷﾒﾃﾞｨｱ¼‰ã ®æ–‡å—åŒ–ã ‘ã «ã ¤ã „ã ¦ã ¯ã€ Help:ç‰¹æ®Šæ–‡å—ã‚’ã ”è¦§ã ã ã •ã „ã€‚

æ–‡å—åŒ–ã ‘ï¼ˆã„ã ¯ã °ã ‘ï¼‰ã ¯ã€ ã‚³ãƒ³ãƒﾟｭｰﾀｰ¿ã §æ–‡å—ã‚’è¡¨ç¤ºã ™ã‚‹és›ã «ã€ æ£ã —ã è¡¨ç¤ºã •ã‚Œã ªã „ç ¾è±¡ã ®ã “ã ¨ã€‚

- ä¾‹ï¼šã€Œæ–‡å—åŒ–ã ‘ã€ ã Œã€ ã€Œ Ã¦â€“â€¡Ã¥Ââ€”Ã¥Å’â€“Ã£Â â€¯ ã€ ã„„ã€Œè ï¿½ï½æ€œå–§ç„¡ï¿½ã€ ã ¨è¡¨ç¤ºã •ã‚Œã‚‹ã ª ©ã€‚

ã€Œæ–‡å—åŒ–ã ‘ã€ ã ¨ã „ã †è¨€è‘‰ã ¯ã€ ã‚³ãƒ³ãƒﾟｭｰﾀｰ¿ç’°å¢ƒ §åŽŸå‰‡ã ¨ã —ã ¦ãƒžãƒ«ãƒ ãƒ ã‚¤ãƒˆæ–‡å—ã‚’ä½¿ç”¨ã —ã ª ã „æ¬§ç±³ç‰ã ®ãƒ©ãƒ†ãƒ³ãƒ«ãƒ•ã‚¡ãƒ™ãƒｯﾄãƒ™ｰｽ ã ®ä½¿ç用è¨€ã ª ã Šã „ã ¦ã ¯è©²å½“ã ™ã‚‹ﾌｫﾝﾄ Œåˆ†ã œ¨ã —ã ª ã £ã Ÿã “ã ¨ã ‹ã‚‰ã€ æ—¥æœ¬è ª ã ® â€œMojibakeâ€ ã ¨ã „ã †è¨€è‘‰ã Œã ã ®ã ¾ã ¾é€šç”¨ã ™ã‚‹ã‚ˆã †ã «ã ª ã £ã Ÿã€‚â†’#Mojibake

ç›®æ¬¡ [é žè¡¨ç¤º]

Mojibake

ウィキペディア
フリー百科事典

ãƒ¡ã‚¤ãƒšãƒ¼ã‚¸
ã‚³ãƒŸãƒ¥ãƒ‹ãƒ†ã‚£ãƒ»ãƒãƒｰﾼã‚¿ãƒ«
æœ€è¿‘ã ®å‡ºæ ¥ä°‹
æ–°ã —ã „ãƒšãƒｰﾼã‚¸
æœ€è¿‘ã ®æ›´æ–°°
ã Šã ¾ã ‹ã ›è¡¨ç¤º
ç·¨ç¿’ç”¨ãƒšãƒｰﾼã‚¸
ã‚¢ãƒƒãƒﾌﾟãƒﾛｰﾄﾞ
(ã‚¡ã‚£ã‚-ãƒ¡ãƒ‡ã‚£ã‚¢ãƒ»ã‚³ãƒ¢ãƒ³ã‚°)

ãƒ¯ãƒ«ãƒﾟ

ãƒ¯ãƒ«ãƒﾟ
ä°•æ^ç«¯
ã ŠçŸ¥ã‚‰ã ›
ãƒ ã‚°ã ®å ±ã‚Š
å¯„ä»˜
ã‚¡ã‚£ã‚-ãƒšãƒ¡ã‚¢ã‚¢ã‚¢ «é–¢ã ™ã‚‹ã ‹ã Šå• ã „å ^ã‚ã ›

ãƒ„ãƒｰﾼ«

ãƒªãƒ³ｸ先ãƒﾍﾟｰｼ

```
└─$ curl -s https://packages.stripe.dev/api/security/keypair/stripe-cli-gpg/public | gpg --dearmor | sudo tee
/usr/share/keyrings/stripe.gpg
[sudo] password for meh:
```

Mojibake

```
└─$ curl -s https://packages.stripe.dev/api/security/keypair/stripe-cli-gpg/public | gpg --dearmor | sudo tee
/usr/share/keyrings/stripe.gpg
[sudo] password for meh:
```

ExpÃ©dition de votre commande

The first encodings

1963 : ANSI : ASCII ! 7 bits → 127 characters

1972 : 8 bits CPUs are here ! 8 bits → 255 characters

1991 : Unicode V1.0 !

- universal
- uniform
- unique

The first encodings

1963 : ANSI : ASCII ! 7 bits → 127 characters

1972 : 8 bits CPUs are here ! 8 bits → 255 characters

1991 : Unicode V1.0 ! **16 bits → 65 536 characters**

- universal
- uniform
- unique

Code points

A’s code point is U+0041

B’s is U+0042

…

Ω’s is U+2126

The first encodings

1963 : ANSI : ASCII ! 7 bits → 127 characters

1972 : 8 bits CPUs are here ! 8 bits → 255 characters

1991 : Unicode V1.0 ! 16 bits → 65 536 characters

1996 : Unicode V2.0 presents **UTF** ! Encoding ≠ Code Point

- UTF-32 → 32 bits, 4 bytes
- UTF-16 → 16 bits, 2 bytes
- UTF-8 → 8 bits, 1 to 4 bytes

A	Ω	語	𐎄	UTF-32
00000041	000003A9	00008A9E	00010384	

A	Ω	語	𐎄	UTF-16
0041	03A9	8A9E	D800 | DF84	

A	Ω	語	𐎄	UTF-8
41	CE | A9	E8 | AA | 9E	F0 | 90 | 8E | 84	

BOM

- Encoded as FE FF (or FF FE if you are swapping high and low bytes)

- Indicates that the text's encoding is Unicode
  - and in which Unicode encoding

- Byte order (endianness) of the text's stream for 16-bits & 32-bits encodings

BOM

green-en.vtt

```
WEBVTT FILE

1
00:00.140 --> 00:02.510
What do I gotta do?
Oh I gotta hit the button,
```

```php
if (strpos($signature, 'WEBVTT') !== 0) {
    $parsing_errors[] = 'Missing "WEBVTT" at the beginning of the file';
}
```

BOM

```
// Strip UTF-8 BOM
$bom = pack('CCC', 0xEF, 0xBB, 0xBF);
if (substr($content, 0, 3) === $bom) {
    $content = substr($content, 3);
}
```

BOM

Handled in the Serializer component :

- stripped when decoding csv

```
$csv = "\xEF\xBB\xBF".<<<'CSV'
foo,bar
hello,"hey ho"
CSV;

$this->encoder->decode($csv, 'csv', [CsvEncoder::AS_COLLECTION_KEY => false]);
// ['foo' => 'hello', 'bar' => 'hey ho']
```

BOM

Handled in the Serializer component :

- stripped when decoding csv

- can be added on demand in the output

```
$this->encoder->encode($value, 'csv', [CsvEncoder::OUTPUT_UTF8_BOM_KEY => true]));
```

✨ UTF-8 ✨

- 8 bits
- 0-127 : 1 byte → Backward compatibility with ASCII
- 128+ : 2 to 6 bytes

Trivia: Replacement character

OBJ U+FFFC

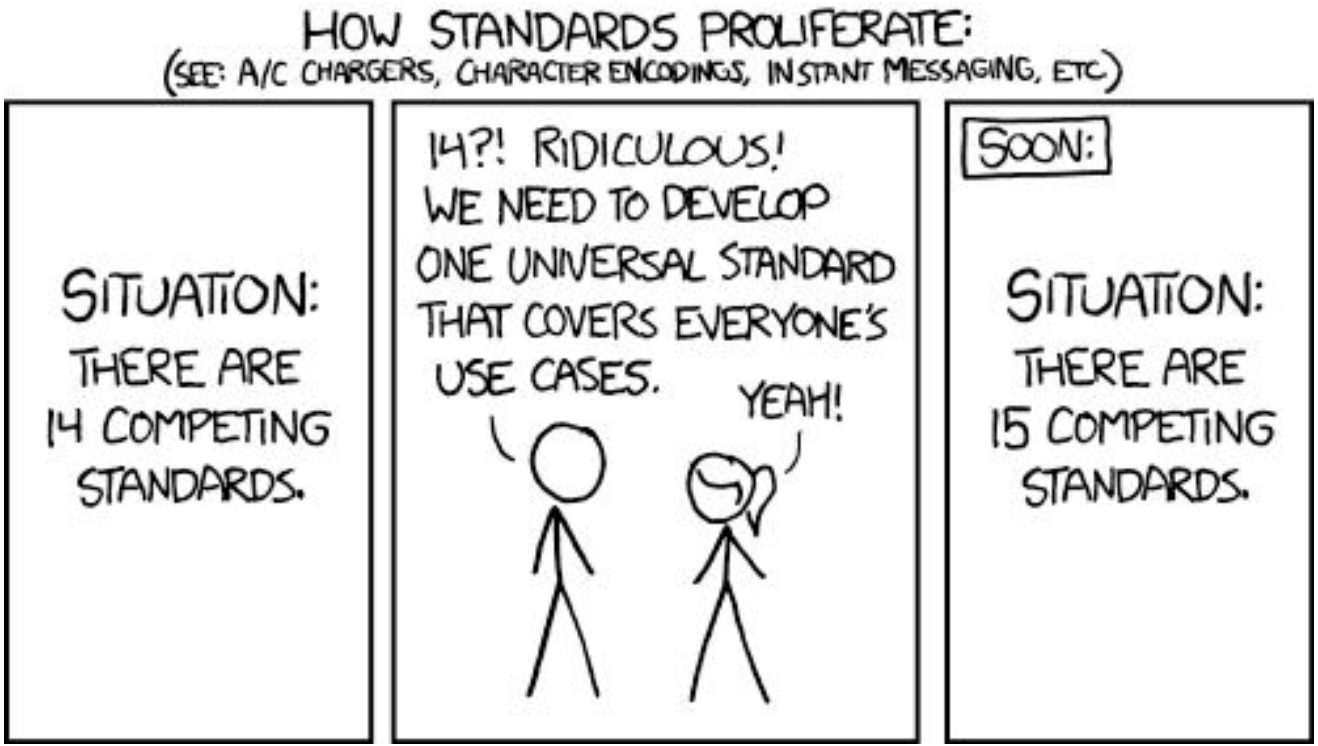
U+FFFD

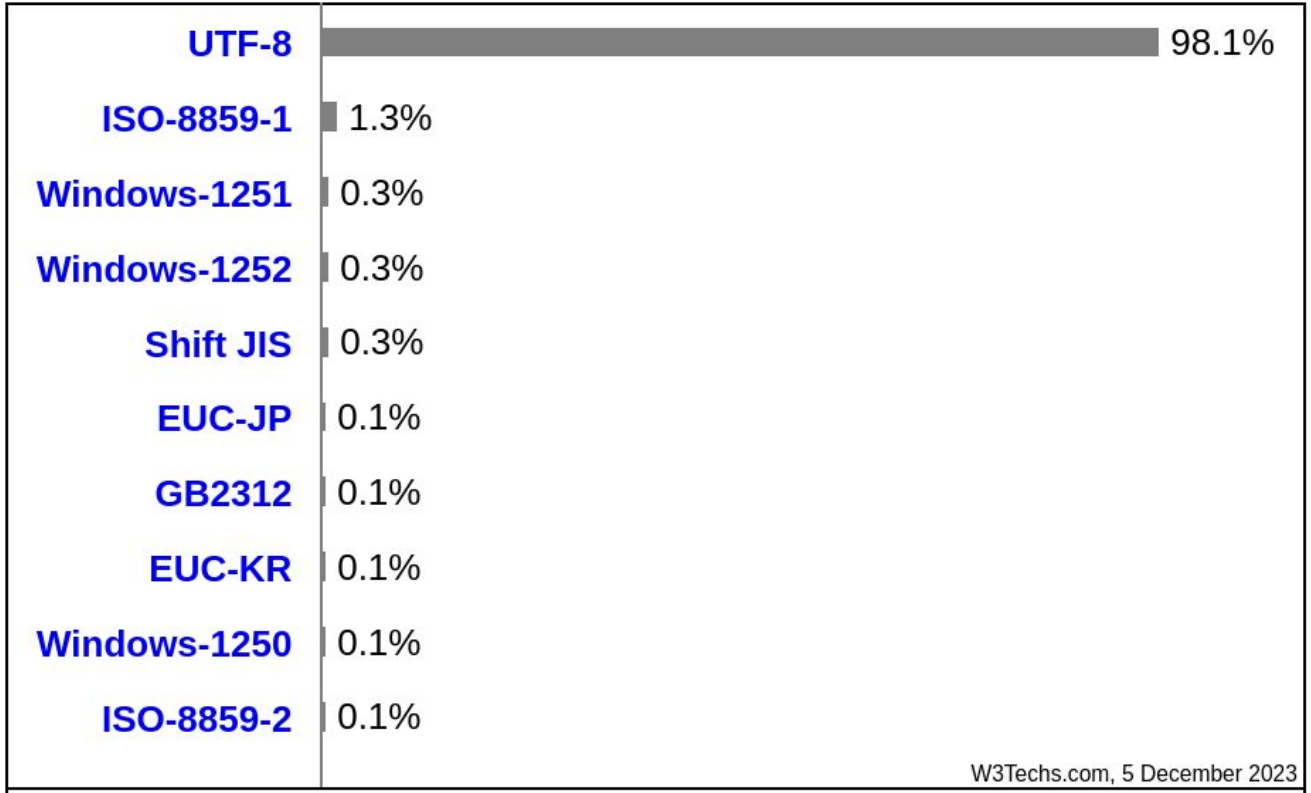
≠

FFFE U+FFFE

FFFF U+FFFF

Standards


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.



Percentages of websites using various character encodings
Note: a website may use more than one character encoding

The first encodings

1963 : ASCII ! 7 bits → 127 characters

1972 : 8 bits CPUs are here ! → 255 characters

1991 : Unicode V1.0 ! 16 bits → 65 536 characters

1991 : Unicode V1.0 ! 16 bits → 65 536 characters

1996 : Unicode V2.0 presents UTF ! Encoding ≠ Code Point

2023 : Unicode V15.0 → 149 186 characters and around 245 000 code points assigned in a space that can contain up to 1 114 112 different code points

To sum it up

Graphemes	C	a	f	é	
Glyphs	C	a	f	e	´
Code points	U+0043	U+0061	U+0066	U+0065	U+0301
UTF-32 Bytes	00 00 00 43	00 00 00 61	00 00 00 66	00 00 00 65	00 00 03 01
UTF-16 Bytes	00 43	00 61	00 66	00 65	03 01
UTF-8 Bytes	43	61	66	65	CC 81

To sum it up

Graphemes	C	a	f	é	
Glyphs	C	a	f	e	´
Code points	U+0043	U+0061	U+0066	U+0065	U+0301
UTF-32 Bytes	00 00 00 43	00 00 00 61	00 00 00 66	00 00 00 65	00 00 03 01
UTF-16 Bytes	00 43	00 61	00 66	00 65	03 01
UTF-8 Bytes	43	61	66	65	CC 81

To sum it up

Graphemes	C	a	f	é	
Glyphs	C	a	f	e	´
Code points	U+0043	U+0061	U+0066	U+0065	U+0301
UTF-32 Bytes	00 00 00 43	00 00 00 61	00 00 00 66	00 00 00 65	00 00 03 01
UTF-16 Bytes	00 43	00 61	00 66	00 65	03 01
UTF-8 Bytes	43	61	66	65	CC 81

To sum it up

Graphemes	C	a	f	é	
Glyphs	C	a	f	e	´
Code points	U+0043	U+0061	U+0066	U+0065	U+0301
UTF-32 Bytes 00	00 00 43	00 00 00 61	00 00 00 66	00 00 00 65	00 00 03 01
UTF-16 Bytes	00 43	00 61	00 66	00 65	03 01
UTF-8 Bytes	43	61	66	65	CC 81

Symfony's String component

Symfony's string component

Provides an object oriented approach to strings manipulation.

```
new UnicodeString('Å');  ──────►  u("Å");
                                  }
new ByteString('Å');     ──────►  b("Å");   s("Å");
```

```
new CodePointString('Å');
```

Symfony's string component

```
$text = (new UnicodeString('This is a déjà-vu situation.'))
    ->trimEnd('.')
    ->replace('déjà-vu', 'jamais-vu')
    ->append('!');
// $text = 'This is a jamais-vu situation!'
```

Symfony's string component

```
u('foo BAR')->upper(); // 'FOO BAR'
u('FOO Bar')->lower();  // 'foo bar'
u('Die O\'Brian Straße')->folded(); // "die o'brian strasse"

u('Foo: Bar-baz.')->camel(); // 'fooBarBaz'
u('Foo: Bar-baz.')->snake(); // 'foo_bar_baz'
u('Foo: Bar-baz.')->camel()->title(); // 'FooBarBaz'
```

Symfony's string component

```
u('abc')->indexOf('B');                // null
u('abc')->ignoreCase()->indexOf('B'); // 1

u('hello')->append('world');       // 'helloworld'
u('hello')->append(' ', 'world'); // 'hello world'

u('User')->ensureEnd('Controller');            // 'UserController'
u('UserController')->ensureEnd('Controller'); // 'UserController'
```

Symfony's string component

```
u(' Lorem Ipsum ')->padBoth(20, '-'); // '--- Lorem Ipsum ----'

u('_.')->repeat(10); // '_._._._._._._._._._.'

u('   Lorem Ipsum   ')->trim(); // 'Lorem Ipsum'

u('http://symfony.com')->replace('http://', 'https://');

u('Symfony is great')->slice(0, 7);  // 'Symfony'

u('template_name.html.twig')->split('.');    // ['template_name', 'html', 'twig']
```

ByteString specific methods

```
// returns TRUE if the string contents
// are valid UTF-8 contents
b('Lorem Ipsum')->isUtf8(); // true
b("\xc3\x28")->isUtf8();    // false
```

CodePointString and UnicodeString specific methods

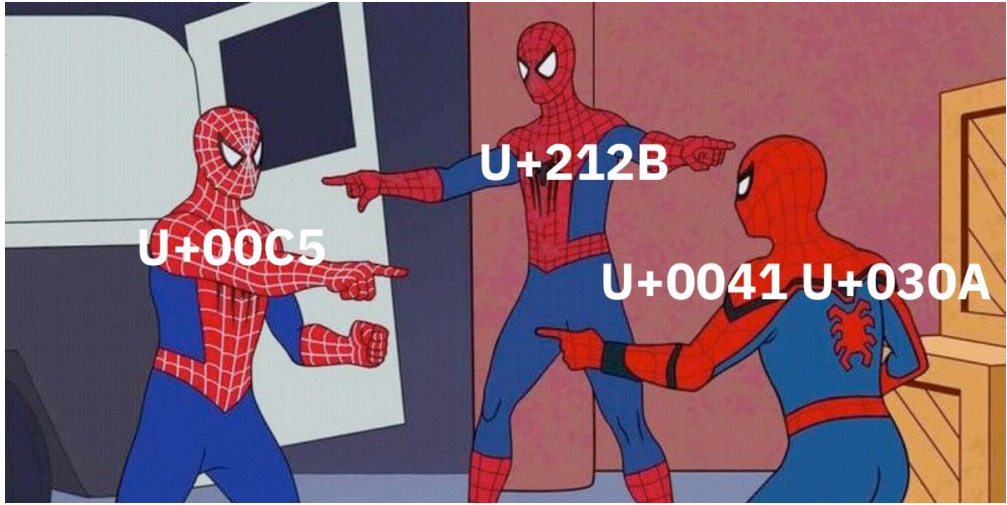
```
u('नमस्ते')->ascii();     // 'namaste'
u('さよなら')->ascii(); // 'sayonara'
u('спасибо')->ascii(); // 'spasibo'

u('नमस्ते')->codePointsAt(0); // न [2344]
u('नमस्ते')->codePointsAt(1); // म [2350]
u('नमस्ते')->codePointsAt(2); // स्ते [2360, 2381, 2340, 2375]
```

Why is "Å" !== "Å" !== "Å"?

Combination of `A` (`U+0041`) and `˚` (`U+030A`)

The codepoint `U+00C5` which gives `Å`, or `“Latin Capital Letter A with Ring Above”`

The codepoint `U+212B` for `“Angstrom Sign Å”`



Normalization

Canonical normalization :

NFD : Canonical Decomposition

Å => A + ˚

NFC : Canonical Composition

A + ˚ => Å

Compatibility normalization :

NFKD : Compatibility Decomposition

Å => A + ˚

NFKC : Compatibility Composition

A + ˚ => Å

Normalization

```
// these encode the letter as a single code point: U+00E5
u('å')->normalize(UnicodeString::NFC);
u('å')->normalize(UnicodeString::NFKC);

// these encode the letter as two code points: U+0061 + U+030A
// a + ◌̊
u('å')->normalize(UnicodeString::NFD);
u('å')->normalize(UnicodeString::NFKD);
```

Normalization

```
$ARing = "\xC3\x85"; // Å (U+00C5)
$ARingComposed = "A"."\xCC\x8A";  // Å (U+030A)

$norm1 = Normalizer::normalize($ARing, Normalizer::FORM_C);
$norm2 = Normalizer::normalize($ARingComposed, Normalizer::FORM_C);

var_dump($ARing === $ARingComposed); // FALSE
var_dump($norm1 === $norm2); // TRUE
```

AsciiSlugger

```
$slugger = new AsciiSlugger();
$slug = $slugger->slug('Wôrķšpáçè ~~sèťtïñğš~~', '/');
// $slug = 'Workspace/settings'
```

AsciiSlugger

```
$slugger = new AsciiSlugger();
$slug = $slugger->slug('Wôrķšpáçè ~~sèťtïñğš~~', '/');
// $slug = 'Workspace/settings'
$slugger = $slugger→withEmoji();

$slug = $slugger→slug('a 😺, and a 🦁 go to 🏞', '-', 'en');
// $slug = 'a-grinning-cat-and-a-lion-go-to-national-park';


$slug = $slugger→slug('un 😺, et un 🦁 vont au 🏞', '-', 'fr');
// $slug = 'un-chat-qui-sourit-et-un-tete-de-lion-vont-au-parc-national';
```

Inflector

```
$inflector = new EnglishInflector();

$result = $inflector->singularize('teeth');        // ['tooth']
$result = $inflector->singularize('radii');        // ['radius']
$result = $inflector->singularize('leaves');       // ['leaf', 'leave', 'leaff']

$result = $inflector->pluralize('bacterium');    // ['bacteria']
$result = $inflector->pluralize('news');         // ['news']
$result = $inflector->pluralize('person');       // ['persons', 'people']
```

Symfony's string component

```
(new ByteString('🤦🏼‍♀️'))->length();        // 17

(new CodePointString('🤦🏼‍♀️'))->length(); // 5

(new UnicodeString('🤦🏼‍♀️'))->length();    // 1
```

CodePointString and UnicodeString specific methods

```
u('👩‍👧‍👧')->codePointsAt(0);
```

```
[
  0 => 128105     U+1F469   👩 WOMAN
  1 => 8205       U+0200D   ZERO WIDTH JOINER
  2 => 128105     U+1F469   👩 WOMAN
  3 => 8205       U+0200D   ZERO WIDTH JOINER
  4 => 128103     U+1F467   👧 GIRL
  5 => 8205       U+0200D   ZERO WIDTH JOINER
  6 => 128102     U+1F466   👦 BOY
]
```

What is the length of “🤦🏼‍♂️”?

What is the length of “🤦🏼‍♀️”?

python3 :

```
>>> len("🤦🏼‍♀️") == 5
True
```

JavaScript :

```
"🤦🏼‍♀️".length == 7
True
```

Rust :

```
"🤦🏼‍♀️".len() == 17
true
```

Elixir :

```
String.length("🤦🏼‍♀️")          // 1
```

Swift :

```
var s = "🤦🏼‍♀️"
print(s.count)                  // 1
print(s.unicodeScalars.count)   // 5
print(s.utf16.count)            // 7
print(s.utf8.count)             // 17
```

What is the length of “🤦🏼‍♀️”?

PHP :

```
strlen('🤦🏼‍♀️');                    // 17
mb_strlen('🤦🏼‍♀️', 'UTF-8');        // 5
```

What is the length of “🤦🏼‍♀️”?

PHP :

```
strlen('🤦🏼‍♀️');                    // 17
mb_strlen('🤦🏼‍♀️', 'UTF-8');        // 5
```

Symfony :

```
u('🤦🏼‍♀️')→length();                // 1
```

What is the length of “🤦🏼‍♀️”?

Unicode scalar	UTF-32 code units	UTF-16 code units	UTF-8 code units	UTF-32 bytes	UTF-16 bytes	UTF-8 bytes
U+1F926 FACE PALM 🤦	1	2	4	4	4	4
U+1F3FC EMOJI MODIFIER FITZPATRICK TYPE-3 🏼	1	2	4	4	4	4
U+200D ZERO WIDTH JOINER	1	1	3	4	2	3
U+2642 MALE SIGN ♂	1	1	3	4	2	3
U+FE0F VARIATION SELECTOR-16	1	1	3	4	2	3
Total	**5**	**7**	**17**	**20**	**14**	**17**

What is the length of “🤦🏼‍♀️”?

```
// Symfony\Polyfill\Intl\Grapheme\Grapheme.php
public static function grapheme_strlen($s)
{
    preg_replace('/'.SYMFONY_GRAPHEME_CLUSTER_RX.'/u', '', $s, -1, $len);

    return 0 === $len && '' !== $s ? null : $len;
}
```

Wrong encoding kills 📱

A Cellphone's Missing Dot Kills Two People, Puts Three More in Jail

By **Jesus Diaz** Published April 21, 2008 | Comments (143)

The life of 20-year-old Emine, and her 24-year-old husband Ramazan Çalçoban was pretty much the normal life of any couple in a separation process. After deciding to split up, the two kept having bitter arguments over the cellphone, sending text messages to each other until one day Ramazan wrote "you change the topic every time you run out of arguments." That day, the lack of a single dot over a letter—product of a faulty localization of the cellphone's typing system—caused a chain of events that ended in a violent blood bath (Warning: offensive language ahead.)

Case of the Turkish i

```
u('i')->upper()->toString(); // I
u('ı')->upper()->toString(); // I
u('I')->lower()->toString(); // i
u('İ')->lower()->toString(); // i.
```


```
u('i')->upper()->codePointsAt(0); // [73]
u('ı')->upper()->codePointsAt(0); // [73]
u('I')->lower()->codePointsAt(0); // [105]
u('İ')->lower()->codePointsAt(0); // [105, 775]
```

Case of the Turkish i

```php
// Symfony\Component\String\AbstractUnicodeString.php
public function lower(): static
{
    $str = clone $this;
    $str->string = mb_strtolower(str_replace('İ', 'i', $str->string), 'UTF-8');

    return $str;
}
```

Case of the Turkish i

> So no, you can’t convert string to lowercase without knowing what language that string is written in.

```
var en_US = Locale.of("en", "US");

var tr = Locale.of("tr");


"I".toLowerCase(en_US); // => "i"

"I".toLowerCase(tr);    // => "ı"


"i".toUpperCase(en_US); // => "I"

"i".toUpperCase(tr);    // => "İ"
```

U+00130
LATIN CAPITAL LETTER I WITH DOT ABOVE

YOUR BROWSER İ

CHARACTER PROPERTIES

CODE POINT	U+00130
NAME	LATIN CAPITAL LETTER I WITH DOT ABOVE
UNICODE1 NAME	LATIN CAPITAL LETTER I DOT
BLOCK	LATIN EXTENDED-A
GENERAL CAT	an uppercase letter
CANONICAL COMBINING CLASS	—
BIDIRECTIONAL CLASS	any strong left-to-right character
MIRRORED CHARACTER IN BIDIRECTIONAL TEXT	false
DECOMPOSITION	0049 0307
UPPERCASE MAPPING	
LOWERCASE MAPPING	0069
TITLECASE MAPPING	
NUMERIC VALUE	—
DECIMAL VALUE	304
UTF-8 HEX VALUE	0xC4B0
UTF-16 HEX VALUE	0x00130
UTF-32 HEX VALUE	0x00000130
XHTML VALUE	İ
VIEWS	5878

Trivia: the flags in Unicode

- No fixed codepoint
- Something once in Unicode stays in it forever
- Flags might become obsolete
- The ISO (International Organisation for Standardization) is the reference with its list of flags recognised by the O.N.U.

without font

🅱🅴 => 🅱🅴, flag Belgium

🇧🇪 => 🇧🇪, flag Belgium

with the right font

Symfony's Intl Component 🌍

Symfony's Intl Component

Provides access to the ICU data:

- Language and Script Names
- Country Names
- Locales
- Currencies
- Timezones
- …

EmojiTransliterator

```
use Symfony\Component\Intl\Transliterator\EmojiTransliterator;

// describe emojis in English
$transliterator = EmojiTransliterator::create('en');
$transliterator->transliterate('Menus with 🍕 or 🍝');
// => 'Menus with pizza or spaghetti'

// describe emojis in Ukrainian
$transliterator = EmojiTransliterator::create('uk');
$transliterator->transliterate('Menus with 🍕 or 🍝');
// => 'Menus with піца or спагеті'
```

EmojiTransliterator

```
use Symfony\Component\Intl\Transliterator\EmojiTransliterator;

// describe emojis in Slack short code
$transliterator = EmojiTransliterator::create('slack');
$transliterator->transliterate('Menus with 🥗 or 🧆');
// => 'Menus with :green_salad: or :falafel:'

// use this to describe emojis in Github short code
$transliterator = EmojiTransliterator::create('github');
```

EmojiTransliterator

```
$reverseTransliterator =
    EmojiTransliterator::create('github', EmojiTransliterator::REVERSE);

$reverseTransliterator
    ->transliterate('Menus with :green_salad: or :falafel:');
    // => 'Menus with 🥗 or 🧆'
```

Taming container environment maintenance: let's go Nix-ing!

Jérôme Vieilledent

Producing and happily maintaining container images: :smiley: :white_check_mark:
Including many up-to-date packages: :muscle:
While still delivering older runtimes: :melting_face: :exploding_head:
Dependency hell is this annoying guest who’s ruining your party and making the situation uncomfortable. You didn’t realize they were invited, but now they’re here breaking everything.
Would it be possible to generate and maintain sane up-to-date environments, with packages having eerie dependencies? The answer is YES, and you don’t have to read the Necronomicon to make that happen. It’s called Nix, and it ain't dark magic…
So, are you ready to Nix?

Delivered in English

Room: **Platform.sh room**

Thursday, December 7, 2023 at 16:20 PM – 16:55 PM

PHP's native functions 🐘

Date added

#	Title	Album	Added by	Date added		
1	Trøllabundin Eivør	Slør	lyrixx	1 hour ago	♥	4:31
2	Bouzouki Hanne & Lore	Bouzouki	lyrixx	17 hours ago		6:04
3	Helheim - Extended Kalandra	Helheim (Extended)	lyrixx	4 days ago		4:38
4	Exit in Darkness MONO, A.A. Williams	Exit in Darkness	lyrixx	4 days ago		6:11
5	Brambles Frank Carter & The Rattlesnakes	Brambles	lyrixx	4 days ago		3:32
6	From the Flame Leprous	Malina	lyrixx	5 days ago		3:51
7	Grotto (feat. Magnus Børmark) Einar Solberg, Magnus Børmark	16	lyrixx	5 days ago		4:57
8	A Sunday Kind Of Love Etta James	At Last!	lyrixx	5 days ago		3:16

Transliteration?

- from a script to another
- Αθήνα → Athena
- You might want to transliterate data before indexing it

```
[
  Antwerpen
  Brussel
  Cannes
  // …
  Zurich
  Αθήνα
]
```

Transliteration?

```
transliterator_transliterate(
    'Any-Latin; Latin-ASCII; Lower()',
    "A æ Übérmensch på høyeste nivå! И я люблю PHP! fi"
);
// "a ae ubermensch pa hoyeste niva! i a lublu php! fi"
```

ASCII: chr() and ord()

```
/**
 * Generate a single-byte string from a number
 * @param int $codepoint : The ascii code.
 * @return string the specified character.
 */
#[Pure]
function chr(int $codepoint): string {}

/**
 * Convert the first byte of a string to a value between 0 and 255
 * @param string $character : A character.
 * @return int<0, 255> the ASCII value as an integer.
 */
#[Pure]
function ord(string $character): int {}
```

utf8_ functions

`utf8_encode()` and `utf8_decode()`

utf8_ functions

`utf8_encode()` and `utf8_decode()`

Only from and to ISO-8859-1 !

Deprecated since PHP 8.2.0

mb_ functions

- Like the PHP’s string fonctions, but on more than one byte
- str_replace works just fine if needle and haystack have the same encoding
- You have to manually enable the **mbstring** extension in PHP

Emojis as class names...

```
interface 🍚 {}
interface 🐟 {}

class 🍣 implements 🍚, 🐟 {

}
```

…or any other Unicode character

```
interface ┻━━┻ {}

class (╯°□°)╯︵┻━━┻ extends Exception implements ┻━━┻ {
  public function __construct($message = __CLASS__, $code = 0, Exception
$previous = null) {
    parent::__construct($message, $code, $previous);
  }
}

class (ノ゜Д゜)ノ︵┻━━┻ extends Exception implements ┻━━┻ {
  public function __construct($message = __CLASS__, $code = 0, Exception
$previous = null) {
    parent::__construct($message, $code, $previous);
  }
}
```

Trivia: kaomojis

¯_(ツ)_/¯

(͡° ͜ʖ ͡°)

(╯°□°)╯ ︵ ┻━┻

(~˘▽˘)~

┬─┬ノ(ಠ_ಠノ)

(凸ಠ益ಠ)凸

More readable mathematics !

```
$√2π = sqrt(2 * $π);
$(z + g +½)^z+½ = pow($z + $g + 0.5, $z + 0.5);
$e^-(z + g +½) = exp(-($z + $g + 0.5));

/**
 * Put it all together:
 *   __  /        1 \ z+½
 *  √2π | z + g + -  |    e^-(z+g+½) A(z)
 *      \        2 /
 */
return $√2π * $(z + g +½)^z+½ * $e^-(z + g +½) * $A(z);
```

Spaces in method names

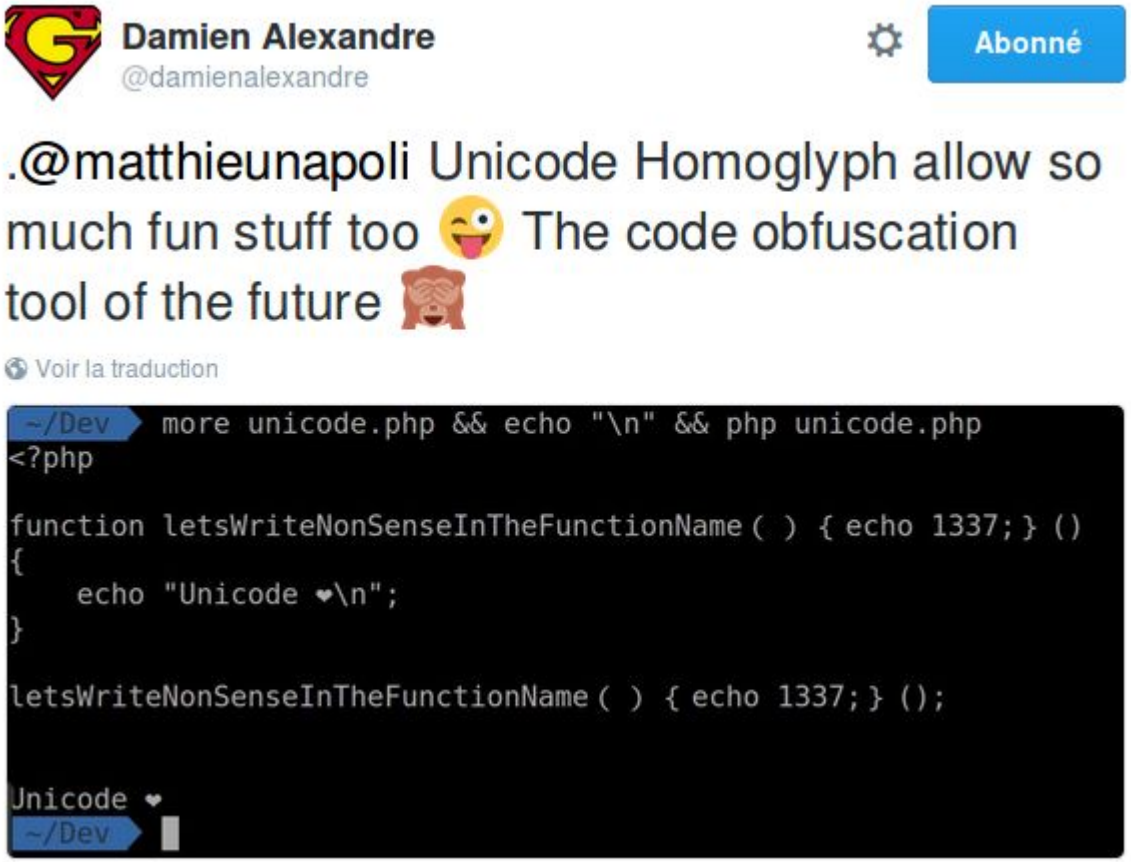
Homoglyphs

(	U+0028	LEFT PARENTHESIS
（	U+FF08	FULLWIDTH LEFT PARENTHESIS
﹙	U+FE59	SMALL LEFT PARENTHESIS
⁽	U+207D	SUPERSCRIPT LEFT PARENTHESIS
₍	U+208D	SUBSCRIPT LEFT PARENTHESIS
❨	U+2768	MEDIUM LEFT PARENTHESIS ORNAMENT

CONFUSABLES

[((((

Homoglyphs


Damien Alexandre
@damienalexandre
Abonné
.@matthieunapoli Unicode Homoglyph allow so much fun stuff too 😜 The code obfuscation tool of the future 🙈
Voir la traduction
~/Dev more unicode.php && echo "\n" && php unicode.php
<?php
function letsWriteNonSenseInTheFunctionName ( ) { echo 1337; } ()
{
echo "Unicode ❤\n";
}
letsWriteNonSenseInTheFunctionName ( ) { echo 1337; } ();
Unicode ❤
~/Dev

Inside your database 💾

Set names, Charset, Collate?

```
SET NAMES utf8mb4 COLLATE utf8mb4_general_ci;
```

```
CREATE DATABASE awesome_database
CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
```

Charsets & Collations

my_alphabet = [

0 = A,

1 = B,

2 = a,

3 = b

]

encoding

glyph

Character set

Charsets & Collations

```
my_alphabet = [
    0 = A,
    1 = B,
    2 = a,
    3 = b
]
```

A ? B

Charsets & Collations

```
my_alphabet = [
    0 = A,
    1 = B,
    2 = a,
    3 = b
]
```

A < B

Charsets & Collations

my_alphabet = [

0 = A,

1 = B,

2 = a,

3 = b

]

Collation

a < b

Example of collations

- utf8mb4_unicode_ci :
  - sorts "ß" like "ss"
  - "Œ" like "OE"
  - Ignorable characters are skipped

- utf8mb4_general_ci : as single characters
  - "ß" like "s”
  - "Œ" like "e"

SET NAMES?

```
SET character_set_client = 'utf8mb4'

SET character_set_connection = 'utf8mb4'

SET character_set_results = 'utf8mb4'
```

Naming of UTF-8

- PostgreSQL : `UTF8`
  - 8 bits
  - 1 to 4 bytes

- Oracle : `AL32UTF8` (Real UTF-8, Unicode 9.0)
  - Not `UTF8` (Actually CESU-8, Unicode 3.0)

- MySQL : `utf8mb4`
  - Not `utf8` (UTF-8 on 3 bytes)

MySQL's "utf8"

2002 : UTF-8 standard would allow up to 6 bytes per character

Speed boost if all rows are the same number of bytes in a table

People would use CHAR because it has a defined number of characters, no matter which value is stored

CHAR(1) = 6 bytes, CHAR(2) = 12 bytes, …

2003 : The old UTF-8 standard is declared obsolete by Unicode to make room to the new one

Will people try to encode their CHAR columns into UTF-8? Let's change the size!

mysql-cluster-8.2.0 ··· mysql-4.1.0

bar@bar.mysql.r18.ru committed on Sep 27, 2002

Showing **1 changed file** with **1 addition** and **1 deletion**.

2 strings/ctype.c

```diff
@@ -3772,7 +3772,7 @@ CHARSET_INFO compiled_charsets[] = {
 3772 3772        my_strnncoll_utf8,  /* strnncoll    */
 3773 3773        my_strnxfrm_utf8,   /* strnxfrm     */
 3774 3774        NULL,               /* like_range   */
-3775      -      6,                  /* mbmaxlen     */
+     3775 +      3,                  /* mbmaxlen     */
 3776 3776        my_ismbchar_utf8,   /* ismbchar     */
 3777 3777        my_ismbhead_utf8,   /* ismbhead     */
 3778 3778        my_mbcharlen_utf8,  /* mbcharlen    */
```

mysql-cluster-8.2.0 ··· mysql-4.1.0

bar@bar.mysql.r18.ru committed on Sep 27, 2002

Showing **1 changed file** with **1 addition** and **1 deletion**.

2 strings/ctype.c

```
@@ -3772,7 +3772,7 @@ CHARSET_INFO compiled_charsets[] = {
3772  3772        my_strnncoll_utf8,  /* strnncoll    */
3773  3773        my_strnxfrm_utf8,   /* strnxfrm     */
3774  3774        NULL,               /* like_range   */
3775        -     6,                  /* mbmaxlen     */
      3775  +     3,                  /* mbmaxlen     */
3776  3776        my_ismbchar_utf8,   /* ismbchar     */
3777  3777        my_ismbhead_utf8,   /* ismbhead     */
3778  3778        my_mbcharlen_utf8,  /* mbcharlen    */
```

10 comments on commit `43a506c`

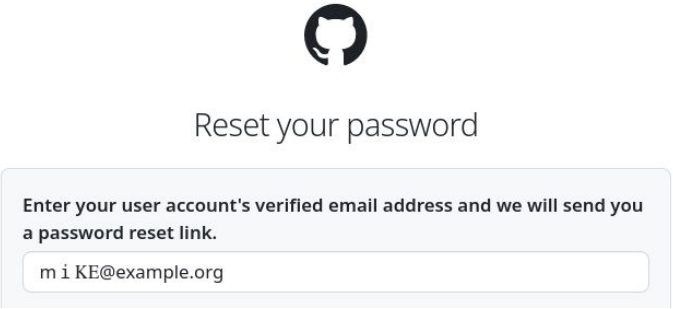
zort commented on `43a506c` on Jan 25, 2018

Wtf why

108 14 1

Security issues 🔓

Homoglyph attack



Password forgotten?

Enter a fake email address, looking like the one you're attacking

`mike@example.org` != `m ı □□@example.org`

The mail will be normalized before looking it up in the database

A token for `mike@example.org` is generated then sent to `m ı □□@example.org` who can now connect as `mike@example.org`

Phabricator

> On inserting Unicode characters with code points greater than `0xFFFF` into columns that have a `utf8` charset. MySQL then truncates a string as soon as it reaches such a character.

Domain restricted subscription

Enter “attacker@gmail.com🍕@allowed-domain.com”

If the check on domain is valid

Only “attacker@gmail.com” is stocked in the DB !

Happy encoding! 🛠️

Thank you!
Questions?

SymfonyCon Brussels 2023 - Marion Hurteau
@marionherisson
mhurteau@jolicode.com

SOURCES – Webpages

https://adamhooper.medium.com/in-mysql-never-use-utf8-use-utf8mb4-11761243e434
https://decodeunicode.org/
https://deliciousbrains.com/how-unicode-works/
https://dev.mysql.com/doc/refman/8.0/en/charset-general.html
https://github.com/brefphp/bref/blob/f4df37277181dc76b6f644663de236eae7a793d2/src/functions.php#L11
https://github.com/captioning/captioning/issues/86
https://github.com/jolicode/emoji-search
https://github.com/markrogoyski/math-php
https://github.com/mysql/mysql-server/commit/43a506c0ced0e6ea101d3ab8b4b423ce3fa327d0
https://github.com/PHP-CS-Fixer/PHP-CS-Fixer/blob/master/src/Fixer/Basic/EncodingFixer.php
https://github.com/sgolemon/table-flip/blob/master/src/TableFlip.php
https://github.com/symfony/symfony/blob/85b97226def5e4a50c1e3805a6c31bb6642efb70/src/Symfony/Component/Intl/Tests/Transliterator/EmojiTransliteratorTest.php
https://github.com/symfony/symfony/pull/33896/files
https://gizmodo.com/a-cellphones-missing-dot-kills-two-people-puts-three-m-382026
https://hackerone.com/reports/2233
https://hsivonen.fi/string-length/
https://www.joelonsoftware.com/2003/10/08/the-absolute-minimum-every-software-developer-absolutely-positively-must-know-about-unicode-and-character-sets-no-excuses/
https://jolicode.github.io/unicode-conf/index.html#/
https://kunststube.net/encoding/
https://news.ycombinator.com/item?id=8892157

https://www.php.net/manual/en/function.utf8-decode.php
https://www.php.net/manual/en/mbstring.supported-encodings.php
https://www.php.net/manual/fr/refs.international.php
https://www.postgresql.org/docs/current/multibyte.html
https://pyrech.github.io/php-wtf/#/15?_k=dyazd4
https://stackoverflow.com/questions/766809/whats-the-difference-between-utf8-general-ci-and-utf8-unicode-ci/
https://symfony.com/blog/new-in-symfony-6-2-better-emoji-support
https://symfony.com/doc/current/components/intl.html
https://symfony.com/doc/current/components/string.html
https://tonsky.me/blog/unicode/
http://www.unicode.org/charts/
http://unicode.org/emoji/charts/emoji-variants.html
https://unicode-org.github.io/icu/userguide/transforms/general/#script-transliteration
https://unicode.org/glossary/
https://fr.wikipedia.org/wiki/Wikip%C3%A9dia:Unicode/Test
https://en.wikipedia.org/wiki/Character_encoding
https://en.wikipedia.org/wiki/UTF-8
https://en.wikipedia.org/wiki/Mojibake
https://en.wikipedia.org/wiki/Byte_order_mark
https://www.youtube.com/watch?v=kaucJce8hhE&t=19s&ab_channel=TheUnicodeConsortium

SOURCES – Images

https://unsplash.com/photos/a-plate-of-cheese-jntQPBIK_sE (Christina Deravedisian)
https://unsplash.com/photos/command-computer-keyboard-key-46T6nVjRc2w (Hannah Joshua)
https://unsplash.com/photos/crt-monitor-turned-off-aiqKc07b5PA (Federica Galli)

SOURCES – Books

Unicode à gogo ! by Design Brouhaha