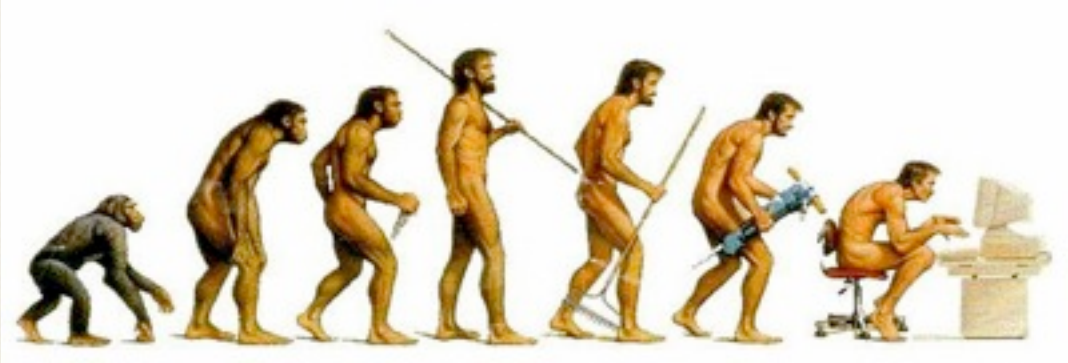
PHP 5.3 == Awesome!

ConFoo 2010 - Ilia Alshanetsky

```
version_compare('5.3.2', '5.2.13', '>')
==
TRUE;
```

EVOLUTIONARY
Evolutionary Release



- Old code will still work
- Focus on mainly improvement of existing functionality
- Fewer bugs
- Faster release cycle

\Namespaces

- Biggest addition in 5.3 code base
- Feature complete implementation of namespaces
- Majority of functionality implemented at compile time
- Simplifies naming conventions

Cleaner Code

Without Namespaces

```
function MY_wrapper() {}

class MY_DB { }

define('MY_CONN_STR', '');



MY_wrapper();

new MY_DB();

MY_CONN_STR;
```

+ =

With Namespaces

```
namespace MY_NS;

function wrapper() {}

class DB { }

const CONN_STR = '';


use MY_NS;

wrapper();

new DB();

CONN_STR;
```

Common Error

```php
<?php
$a = 1;
namespace ns;
```

Fatal error: Namespace declaration statement has to be the very first statement in the script

Multiple Namespaces Per File

```
namespace LIB;

class SQLite {}

$b = new SQLite();

namespace LIB_EXTRA;

class Mcrypt {}

$a = new Mcrypt();

var_dump(
        get_class($b),
        get_class($a)
);
```

```
string(11) "LIB\SQLite"
string(18) "LIB_EXTRA\Mcrypt"
```

Multiple Namespaces Per File
The “Better” Syntax

```
namespace LIB {
  class MySQL {}
  class SQLite {}
}

namespace LIB_EXTRA {
  class Mcrypt {}
}

namespace {
  $b = new SQLite();
  $a = new MScrypt();
}
```

→ Global scope (namespace)

namespace {} tricks

```php
<?php
namespace {
   $a = 1;
}

namespace foo {
   function bar() {}
}
```

Code above a namespace

```php
<?php
namespace {
   $a = 1;
}
namespace foo;
function bar() {}
```

Won't work

Namespace Hierarchy

```
namespace foo;

function strlen($foo) { return \strlen($foo) * 2; }

echo strlen("test"); // 8
echo \strlen("test"); // 4
echo \foo\strlen("test"); // 8
```

- Functions, classes and constant references inside a namespace refer to namespace first and global scope later.

Words of Caution

```
<?php
namespace foo;

function strlen($foo) { return strlen($foo) * 2; }

echo strlen("test");
```

Don’t overwrite native functions unless absolutely necessary and if you do, use CAUTION!

Namespaces & autoload

```
function __autoload($var) { var_dump($var); } // LIB\foo
require "./ns.php"; /*
        <?php
                namespace LIB;
                new foo();
*/
```

- __autoload() will be passed namespace name along with class name.
- autoload will only be triggered if class does not exist in namespace and global scope
- __autoload() declared inside a namespace will not be called!

Other NS Syntactic Sugar

```
namespace really\long\pointlessly\verbose\ns;

__NAMESPACE__; // current namespace name

class a{}

get_class(new a()); // really\long\pointlessly\verbose\ns\a

use really\long\pointlessly\verbose\ns\a AS b;
```

Alias class from a namespace

Improved Performance

- md5() is roughly 10-15% faster
- Better stack implementation in the engine
- Constants moved to read-only memory
- Improve exception handling (simpler & less opcodes)
- Eliminate open(2) call on (require/include)_once
- Smaller binary size & startup size with gcc4
- Substantial performance improvements on Windows (40% +)

Overall Improvement 5-15%

New Language Features

__DIR__

- Introduced __DIR__ magic constant indicating the directory where the script is located.

```
echo dirname(__FILE__); // < PHP 5.3

    /* vs */

echo __DIR__; // >= PHP 5.3
```

?: Operator

- Allows quick retrieval of a non-empty value from 2 values and/or expressions

```
$a = true ?: false; // true
$a = false ?: true; // true
$a = "" ?: 1; // 1
$a = 0 ?: 2; // 2
$a = array() ?: array(1); // array(1);
$a = strlen("") ?: strlen("a"); // 1
```

__callStatic()

- __call() equivalent, but for static methods.

```
class helper {
        static function __callStatic($name, $args) {
                echo $name.'('.implode(',', $args).')';
        }
}

helper::test("foo","bar"); // test(foo,bar)
```

  - Dynamic static function/method calls are kinda slow...

Dynamic Static Calls

- PHP now allows dynamic calls to static methods

```
class helper {
        static function foo() { echo __METHOD__; }
}

$a = "helper";
$b = "foo";

$a::$b(); // helper::foo
```

  - Dynamic static function/method calls are kinda slow...

Late Static Binding

- Processing of static events has been extended into execution time from compile time.

```
class A {
   public static function whoami() {
      echo __CLASS__;
   }

   public static function identity() {
     self::whoami();
   }
}

class B extends A {
   public static function whoami() {
      echo __CLASS__;
   }
}

B::identity(); // A <-- PHP < 5.3
```

```
class A {
   public static function whoami() {
      echo __CLASS__;
   }

   public static function identity() {
      static::whoami();
   }
}

class B extends A {
   public static function whoami() {
      echo __CLASS__;
   }
}

B::identity(); // B <-- >= PHP 5.3
```

- Beware if you use an opcode cache
- Not backwards compatible

Closures

```
$values = array(1,9,4,2);

usort(
    $values,
    function ($a, $b) {
        return strcmp($a, $b);
    }
);
```

- ✳ Greater code clarity

GOTO

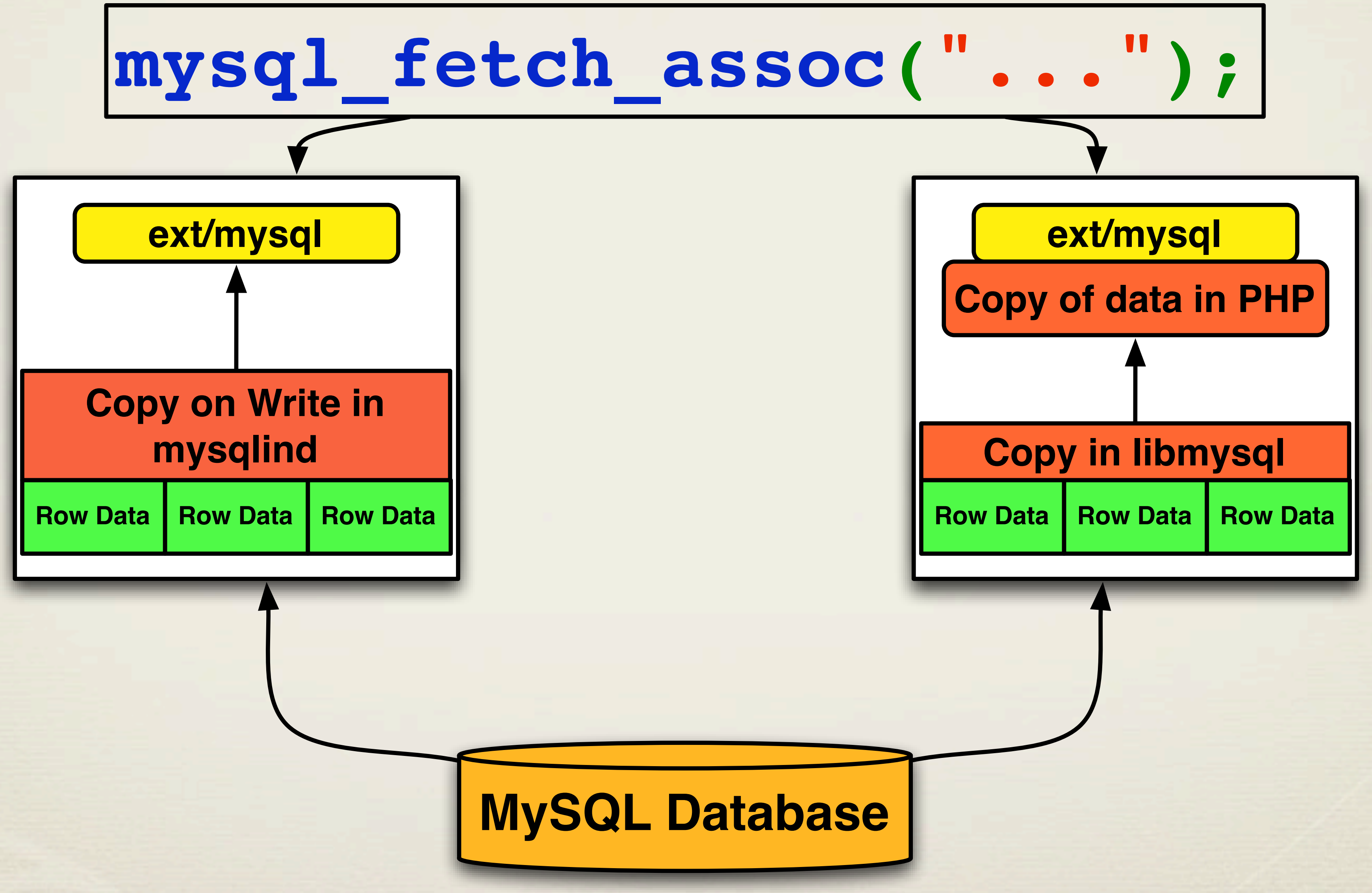
```
if (error_condition1) {
    goto error;
}
if (error_condition2) {
    goto error;
}

error:
    report_error();
    exit;
```

- My favourite feature

MySQLInd

- Specialized, high speed library to interface with MySQL designed specifically for PHP



MySQLInd

- Better performance
- Improved memory usage
- Ability to fetch statistics for performance tuning
- Built-in driver (no external decencies once again)
- Many future options due to tight integration with PHP
- No PDO_MySQL support yet, mysql(i) only for now

INI Magic

- Support for ".htaccess" style INI controls for CGI/FastCGI
- Per-directory INI settings inside php.ini via [PATH=/var/www/domain.com] not modifiable by the user.
- Improve error handling
- Allow use of INI variables and constants from virtually everywhere in the INI files
- Several other minor improvements

INI Magic

```
Name for user-defined php.ini (.htaccess) files. Default is ".user.ini"
user_ini.filename = ".user.ini"

To disable this feature set this option to empty value
user_ini.filename =

TTL for user-defined php.ini files (time-to-live) in seconds.
Default is 300 seconds (5 minutes)
user_ini.cache_ttl = 300

[PATH=/var/www/domain.com]
variables_order = GPC
safe_mode = 1

[my variables]
somevar = “1234”
anothervar = ${somevar} ; anothervar == somevar

[ini arrays]
foo[bar] = 1
foo[123] = 2
foo[] = 3
```

Extra OpenSSL Functions

- Access to OpenSSL Digest Functions

```
foreach (openssl_get_md_methods() as $d) {// MD4, MD5, SHA512... (12 all in all)
  echo $d." - ".openssl_digest("foo", "md5"); //acbd18db4cc2f85cedef654fccc4a4d8
}
```

- Access to OpenSSL Cipher Functions

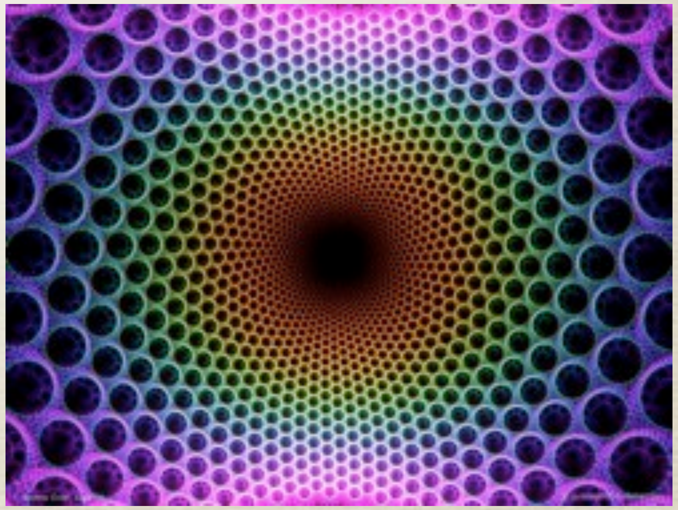
```
// BF-CBC, AES-256 CFB1... (54 all in all)
foreach(openssl_get_cipher_methods() as $v) {
  $val = openssl_encrypt("value", $v, "secret");
  openssl_decrypt($val, $v, "secret"); // value
}
```

- Extend openssl_pkey_new() and openssl_pkey_get_details() functions to allow access to internal DSA, RSA and DH keys.

The goal was to simplify OpenID implementation in PHP

SPL Improvements

- Improve nested directory iterations via FilesystemIterator
- Introduced GlobIterator
- Various data structure classes: SplDoublyLinkedList, SplStack, SplQueue, SplHeap, SplMinHeap, SplMaxHeap, SplPriorityQueue
- Several other tongue twister features



Date Extension Additions

- Controllable strtotime() via date_create_from_format()

```
$date = strtotime("08-01-07 00:00:00");
var_dump(date("Y-m-d", $date)); // string(10) "2008-01-07"
$date = date_create_from_format("m-d-y", "08-01-07");
var_dump($date->format('Y-m-d')); // string(10) "2007-08-01"
```

- Added date_get_last_errors() that returns errors and warnings in date parsing.

```
array(4) {
  ["warning_count"] => int(0)
  ["warnings"] => array(0) { }
  ["error_count"] => int(2)
  ["errors"]=>
  array(2) {
    [2]=> string(40) "The separation symbol could not be found"
    [6]=> string(13) "Trailing data"
  }
}
```

getopt() Improvements

- Works on Windows
- Native implementation not dependent on native getopt() implementation.
- Cross-platform support for longopts (--option)

```
// input: --a=foo --b --c
var_dump(getopt("", array("a:","b::","c")));
/* output: array(3) {
  ["a"]=>
  string(3) "foo"
  ["b"]=>
  bool(false)
  ["c"]=>
  bool(false)
} */
```

XSLT Profiling

- Introduction of Xslt Profiling via **setProfiling()**

```
$xslt = new xsltprocessor();
$xslt->importStylesheet($xml);
$xslt->setProfiling("/tmp/profile.txt");
$xslt->transformToXml($dom);
```

Resulting In:

```
number               match                  name       mode  Calls Tot 100us Avg

    0                 date                                       5       58    11

                         Total                                   5       58
```

E_DEPRECATED

- What would a PHP release be without a new error mode? *deprecated*
- Used to designate deprecated functionality that maybe, sometime in a far future will get removed, maybe.
- Part of E_ALL

Garbage Collector

- Memory cleanup for Über-complex and long scripts that need to free-up memory before the end of execution cycle. (*framework folks, this is for you*)

```
gc_enable(); // Enable Garbage Collector

var_dump(gc_enabled()); // true

var_dump(gc_collect_cycles()); // # of elements cleaned up

gc_disable(); // Disable Garbage Collector
```

NOWDOC

- A HEREDOC where no escaping is necessary

HEREDOC

```
$foo = <<<ONE
this is $fubar
ONE;

/* string(10) "this is" */
```

NOWDOC

```
$bar = <<<‘TWO’
this is $fubar
TWO;

/* string(16) "this is $fubar" */
```

Miscellaneous Improvements

- SQLite Upgraded to 3.6.22
- Hundreds of bug fixes & improvements
- CGI/FastCGI SAPI Improvements
- Various stream improvements
- More things to come ...

Thank You For Listening

Questions?

These slides can be found on: http://ilia.ws
Joind.In: http://joind.in/1272