PHP Version Inspection

Migrating from PHP 7.1.x to 7.2.x:

Introduction to Features:

1. **New Object Type:** A new type, object, has been introduced that can be used for (contravariant) parameter typing and (covariant) return typing of any objects.
2. **Extension loading by name:** Shared extensions no longer require their file extension (.so for Unix or .dll for Windows) to be specified.
3. **Abstract Method Override:** Abstract methods can be override when an abstract class extends another abstract class
4. **Sodium Extension:** Sodium cryptography library became a core extension in PHP
5. **Argon2 Hashing:** introduced in 7.2.x for API hashing
6. **PDO string types extended :**
    - PDO::PARAAM_STR_NATL
    - PDO::PARAM_STR_CHAR
    - PDO::ATTR_DEFAULT_STR_PARAM
7. **Adding address to Sockets Extension:**
    - socket_addrinfo_lookup()
    - socket_addrinfo_connect()
    - socket_addrinfo_bind()
    - socket_addrinfo_explain()
8. **Windows supports proc_nice()**
9. **Endian Support:**
    - pack()
    - unpack()
10. **SQLITE3 can write BLOBS:** allows to open BLOB fields in write mode in only read mode
11. **Oracle OCIB transparent application failover callbacks**
12. **ZIP Extension:** Read/write supports for encrypted archives

Introduction to new Functions:

1. **PHP core**
    - **stream_isatty()**
    - **sapi_windows_vt100_support()**
2. **spl_object_id()**
3. **ftp_append()**
4. **hash_hmac_algos()**
5. **Sockets:**
    - **socket_addrinfo_lookup()**
    - **socket_addrinfo_connect()**
    - **socket_addrinfo_bind()**
    - **socket_addrinfo_explain()**
6. **Multiple Sodium Functions**
7. **ZIP:**
    - **ZipArchive::count()**
    - **ZIpArchive::setEncryptionName()**
    - **ZipArchive::setEncryptionIndex()**

Deprecated features in PHP 7.2.x

1. **Unquoted strings**
2. **png2wbmp() and jpeg2wbmp()**
3. **_autoload()**
4. **create_function()**
5. **parse_str()**
6. **gmp_random()**
7. **each()**
8. **read_exif_data()**

Migrating from PHP 7.2.x to 7.3.x:

Introduced new features:

1. Flexible heredoc and nowdoc syntax
2. Array destructuring supports reference assignments
3. Instance of operator accepts literals
4. Argon2id Supported
5. FastCGI Process Manager
    - Log_limit
    - Log_buffering
    - decorate_workers_output
6. BC Math function
7. Lightweight Directory Access protocol
8. Multibyte String Functions
9. Full case mapping and Case folding support
10. Case insensitive string operations use case folding
11. Unicode 11 Supported
12. Long String Support

Introduced to new functions:

1. PHP Core:
    - array_key_first()
    - array_key_last()
    - gc_status()
    - hrtime()
    - is_countable()
    - net_get_interfaces()
2. fpm_get_status()
3. DateTime::createFromimmutable()
4. GNU multiple Precision:
    - gmp_binomial()
    - gmp_kronecker()
    - gmp_lcm()
    - gmp_perfect_power()
5. openssl_pkey_derive()

6. Sockets
    - socket_wsaprotocol_info_export()
    - socket_wsaprotocol_info_import()
    - socket_wsaprotocol_info_release()

Deprecated Features in 7.3.x

1. Case insensitive Constants
2. Namespaced assert()
3. Searching Strings for non-string Needle
4. Strip-Tags Streaming
5. Data Filtering
    - FILTER_FLAG_SCHEME_REQUIRED
    - FILTER_FLAG_HOST_REQUIRED
6. Image processing and GD
7. Internationalization Functions
8. Multibyte String
9. ODBC and DB2 Functions (PDO_ODBC)

Migrating from PHP 7.3.x to 7.4.x:

Introduced to new features:

1. Type properties (int , string, float, double… etc)
2. Arrow functions
3. Unpacking inside arrays
4. Numeric literal separator
5. Allowing exceptions from __toString() function
6. CURL
7. Filter:
    - FILTER_VALIDATE_FLOAT supports min_range and max_range
8. FFI
9. GD
10. Hash

11. Multibyte String
12. OPcache
13. PDO: username and password can now be specified as part of the PDO DSN for the mysql, mssql, sybase, dblib, firebird and oci drivers
14. PDO_SQLite: allows checking whether the statement is read-only
15. SQLite3: added SQLite3::lastExtendedErrorCode() to fetch last extended result code

New Class and Interface:

1. ReflectionReference: this class provides information about a reference

Introduced new functions:

1. PHP core:
    - get_mangled_object_vars()
    - password_algos()
    - sapi_windows_set_ctrl_handler()
2. imagecreatefromtga()
3. mb_str_split()
4. openssl_x509_verify()
5. pcntl_unshare()
6. SQLite3::backup()
7. SQLite3Stmt::getSQL()

Deprecated Features:

1. Nested ternary operators without explicit parentheses
2. Array and String offset access using curly braces
3. Allow_url_include INI option
4. Invalid characters in base conversion functions
5. Using array_key_exists()
6. hebrevc()
7. convert_cyr_string()
8. money_format()
9. ezmlm_hash()

10. restore_include_path()
11. Implode: passing parameters to the function in reverse order is deprecated. Use implode($glue , $parts) instead of impede ($parts , $glue)
12. FILTER_SANITIZE_MAGIC_QUOTES is deprecated, instead use FILTER_SANITIZE_ADD_SLASHES
13. ReflectionType::__toString() is deprecated.

Removed Extensions:

1. Firebird/Interbase
2. Recode
3. WDDX

Migrating from PHP 7.4.x to 8.0.x:

This major update brings many changes in PHP updates in the functions and features

Introduced to new features:

1. Named Arguments: myFunction(paramNameL $value);
2. Attributes
3. Constructor Property Promotion
4. Union Types
5. Match expression
6. Nullsafe Operator: (?->) supports now
7. Tokenizer: PHPToken adds an object based interface to the tokenizer.

Deprecated Features

1. PHP core:
    - Before: function test($a= [ ], $b) { }
    - After: function test($a, $b) { }
    - function test(?A $a, $b) { } (recommend) but still allows function test (A $a = null, $b) { }

2. **echant_broker_set_dict_path()** and **enchsant_broker_get_dict_path()** are removed because that functionality is neither available in libenchant < 1.5 nor in libenchant-2
3. **echant_dict_add_to_personal()** is deprecated, instead use **enchante_dict_add()**
4. ZIP: when an empty file is zipped with ZipArchive is not supported anymore. Libzip 1.6.0 does not accept empty files as valid zip
5. Procedural API of ZIp is deprecated
6. ReflectionFunction::isDiabled() is deprecated

Backward Incompatible Changes

1. String to Number Comparison
    a.

Comparison	Before	After
`0 == "0"`	`true`	`true`
`0 == "0.0"`	`true`	`true`
`0 == "foo"`	`true`	`false`
`0 == ""`	`true`	`false`
`42 == " 42"`	`true`	`true`
`42 == "42foo"`	`true`	`false`

Other Incompatible Changes:

1. Match is now a reserved keyword
2. Methods with the same name as the class are no longer interpreted as constrictors. __construct() is recommended to use
3. The (real) and (unset) casts have been removed

4. create_function() has been removed
5. each() has been removed instead use foreach() or Arrayiterator
6. Inheritance errors due to incompatible method signatures will now always generate a fatal error
7. mktime() and gmmktime() now require at least one argument.
8. time() function can be use to get the current timestamp
9. image2wbmp() has been removed

Migrating from PHP 8.0.x to 8.1.x:

Introduced to new features;

1. Array Unpacking with String Keys:
    - <?php
    - $arr1 = [1, 'a' => 'b'];
    - $arr2 = [...$arr1, 'c' => 'd']; //[1, 'a' => 'b', 'c' => 'd']
    - ?>
2. Named Argument After Argument Unpacking: Named arguments after an argument unpack is possible in this version e.g. foo(....$args, named: $args)
3. Full path key for File Uploads
4. Readonly property added
5. Hash functions are now supported for the following functions :
    - hash()
    - hash_file()
    - hash_init()

Introduced to new Classes and Interfaces

1. CURLStringFile
2. IntlDatePatternGenerator
3. ReflectionFiber

Introduced to new functions:

1. array_is_list()
2. GD:
    - imagecreatefromavif()
    - imageavif()
3. MySQLi:
    - mysqli_result::fetch_column()
    - mysqli_fetch_column()
    - fsync()
4. Standard:
    - fsync()
    - fdatasync()

Deprecated Features:

1. Implementing serializable without __serialize() and __unserialize()
2. Passing null to non-nullable parameters of built-in functions
3. Implicit incompatible float to int conversions
4. Hash:
    - mhash()
    - mhash_keygen_s2k()
    - mhash_count()
    - mhash_get_block_size()
    - mhash_get_hash_name()
5. GD:
    - imagepolygon()
    - imageopenpolygon()
    - imagefilledpolygon()
6. PDO::FETCH_SERIALIZE has been deprecated

OOP PHP Inspection

OOP stands for **object oriented programming.**

- ☐ Faster and easier to execute
- ☐ Provides a clear structure for the programs
- ☐ DRY - “Don’t Repeat Yourself” this makes the code shorter and reusable

Class and Objects:

These two are the main aspects of OOP
Class is a template for objects
Objects are the instances of a class

Ex.

```php
<?php
class Fruit {
  // Properties
  public $name;
  public $color;

  // Methods
  function set_name($name) {
    $this->name = $name;
  }
  function get_name() {
    return $this->name;
  }
}

$apple = new Fruit();
$apple->set_name('Apple');
echo $apple->get_name();

?>
```

Constructor and Destructor:

1. __contruct(): this method allows to initialize an object's properties when an object is created. This function automatically calls when an object is created. We can pass through multiple parameters to initialize an object's properties
2. __destruct(): this method is automatically call at the end of the script

Access Modifier :

1. Public: this property or method can be accessed from everywhere
2. Protected: this property or method can only be accessed within the class and the class derived from the class
3. This property can only be accessed through this class

Inheritance:

This property derives from another class in OOP. The child class will inherit all the public and protected attributes or methods from the parent class. To inherit a class the keyword '**extends**' is use Ex.

```
<?php
class Fruit {
  public $name;
  public $color;
  public function __construct($name, $color) {
      $this->name = $name;
      $this->color = $color;
  }
  public function intro() {
    echo "The fruit is {$this->name} and the color is
{$this->color}.";
  }
}

// Strawberry is inherited from Fruit
```

```php
class Strawberry extends Fruit {
  public function message() {
    echo "Am I a fruit or a berry? ";
  }
}
$strawberry = new Strawberry("Strawberry", "red");
$strawberry->message();
$strawberry->intro();
?>
```

Abstract Class and Methods:

The purpose of an abstract class is **to function as a base for subclasses**. An abstract class or method is when the parent class has a named method but its child class to fill out the tasks.

These methods or class is defined with the **abstract** keyword

When a child class is inherited from an abstract class, the following are to be followed

- The child class method must be defined with the same name and it redeclares the parent abstract method
- The child class method must be defined with the same or a less restricted access modifier
- The number of required arguments must be the same. However, the child class may have optional arguments in addition

Interface:

Interfaces allow you to specify what methods a class should implement. To implement an interface a class must use the **implements** keyword.

```php
<?php
interface Animal {
  public function makeSound();
}

class Cat implements Animal {
  public function makeSound() {
    echo "Meow";
  }
}

$animal = new Cat();
$animal->makeSound();
?>
```

Static Methods

These methods can be called directly without creating an instance of the class first. Static methods are declared with the **static** keyword. ex.

```php
<?php
class greeting {
  public static function welcome() {
    echo "Hello World!";
  }
}

// Call static method
greeting::welcome();
?>
```

A static method can also be called from a method inside a class:

```php
<?php
class greeting {
  public static function welcome() {
    echo "Hello World!";
  }

  public function __construct() {
     self::welcome();
  }
}

new greeting();
?>
```

Namespace

- They allow for better organization by grouping classes that work together to perform a task
- They allow the same name to be used for more than one class

PHP Iterables

An iterable is any value that can be looped through with a **foreach()** loop. This concept was introduced in php 7.1

```php
<?php
function printIterable(iterable $myIterable) {
  foreach($myIterable as $item) {
    echo $item;
  }
}

$arr = ["a", "b", "c"];
printIterable($arr);
?>
```

Composer

Composer is a tool for dependency management in PHP. It allows you to declare the libraries your project depends on and it will manage (install/update) them for you.

This is not a package manager, it deals with packages or libraries. Composer manages the packages or libraries inside a project. Since it does not install anything globally therefore it is a dependency manager.

System Requirements: composer latest version requires PHP 7.2.5 to run
Installation:

locally:

```
php composer-setup.php --install-dir=bin --filename=composer
```

Globally:

```
mv composer.phar /usr/local/bin/composer
```

How it works:

Composer uses Packagist.org as a main bundles provider. It provides files from repositories that users report on the site. Packagist.org hands over such features versioning or the integration with Github or Bitbucket

Why we need Composer:

Our projects depend on a number of libraries and some of these depend on other libraries. Composer enables you to declare the libraces you depend on. Composer finds which libraries are only required to run the project and only install them.