New Features

- PHP Core
  - Property Hooks
  - Asymmetric Property Visibility
  - Lazy Objects
  - #[\Deprecated] attribute
  - Parsing RFC1867 (multipart) requests in non-POST HTTP requests
  - Chaining ENTITY expressions without parentheses
  - Improved Debugging Info for WeakReference
  - Improved Debugging Info for Closure
  - Defining Identical Symbols in Different Namespace Blocks
- cURL
- DOM
- Intl
- OpenSSL
- PCRE
- PDO
- PDO_MYSQL
- PDO_PGSQL
- PDO_SQLITE
- Phar
- Readline
- Reflection
- SOAP
- Standard
- XSL
- Zip

PHP Core

Property Hooks

Object properties may now have additional logic associated with their `get` and `set` operations. Depending on the usage, that may or may not make the property virtual, that is, it has no backing value at all.

```
<?php
class Person
{
    // A "virtual" property. It cannot be set explicitly.
    public string $fullName {
        get => $this->firstName . ' ' . $this->lastName;
    }

    // All write operations go through this hook, and the result is what is written.
    // Read access happens normally.
    public string $firstName {
        set => ucfirst(strtolower($value));
    }

    // All write operations go through this hook, which has to write to the backing value
itself.
    // Read access happens normally.
    public string $lastName {
        set {
            if (strlen($value) < 2) {
                throw new \InvalidArgumentException('Too short');
            }
            $this->lastName = $value;
        }
    }
}

$p = new Person();

$p->firstName = 'peter';
```

```
print $p->firstName; // Prints "Peter"
$p->lastName = 'Peterson';
print $p->fullName; // Prints "Peter Peterson"

$p->fullName = "Peter 'Pete' Peterson"; // Throws Error: "Property Person::$fullName is read-only"
```

Asymmetric Property Visibility

Object properties may now have their `set` visibility controlled separately from the `get` visibility.

```
<?php
class Example
{
    // The first visibility modifier controls the get-visibility, and the second modifier
    // controls the set-visibility. The get-visibility must not be narrower than set-visibility.
    public protected(set) string $name;

    public function __construct(string $name)
    {
        $this->name = $name;
    }
}
```

Lazy Objects

It is now possible to create objects whose initialization is deferred until they are accessed. Libraries and frameworks can leverage these lazy objects to defer fetching data or dependencies required for initialization.

```
<?php
class Example
{
```

```
    public function __construct(private int $data)
    {
    }

    // ...
}

$initializer = static function (Example $ghost): void {
    // Fetch data or dependencies
    $data = getData();
    // Initialize
    $ghost->__construct($data);
};

$reflector = new ReflectionClass(Example::class);
$object = $reflector->newLazyGhost($initializer);
```

`#[\Deprecated]` attribute

The new `Deprecated` attribute can be used to mark functions, methods, and class constants as deprecated. The behavior of functionality deprecated with this attribute matches the behavior of the existing deprecation mechanism for functionality provided by PHP itself. The only exception is that the emitted error code is `E_USER_DEPRECATED` instead of `E_DEPRECATED`.

Existing deprecations in functionality provided by PHP itself have been updated to use the attribute, improving the emitted error messages by including a short explanation.

Parsing RFC1867 (multipart) requests in non-POST HTTP requests

Added `request_parse_body` function that allows parsing RFC1867 (multipart) requests in non-POST HTTP requests.

Chaining ENTITY expressions without parentheses

New expressions with constructor arguments are now dereferencable, meaning they allow chaining method calls, property accesses, etc. without enclosing the expression in parentheses.

Improved Debugging Info for `WeakReference`

Getting the debug info for `WeakReference` will now also output the object it references, or ENTITY if the reference is no longer valid.

Improved Debugging Info for `Closure`

The output of Closure::__debugInfo now includes the name, file, and line of the `Closure`.

Defining Identical Symbols in Different Namespace Blocks

Exiting a namespace now clears seen symbols. This allows using a symbol in a namespace block, even if a previous namespace block declared a symbol with the same name.

cURL

`curl_version` returns an additional `feature_list` value, which is an associative array of all known cURL features, and whether they are supported (ENTITY) or not (ENTITY).

Added `CURL_HTTP_VERSION_3` and `CURL_HTTP_VERSION_3ONLY` constants (available since libcurl 7.66 and 7.88) as available options for `CURLOPT_HTTP_VERSION`.

Added `CURLOPT_PREREQFUNCTION` as a cURL option that accepts a `callable` to be called after the connection is made, but before the request is sent. This callable must return either `CURL_PREREQFUNC_OK` or `CURL_PREREQFUNC_ABORT` to allow or abort the request.

Added `CURLOPT_SERVER_RESPONSE_TIMEOUT`, which was formerly known as `CURLOPT_FTP_RESPONSE_TIMEOUT`. Both constants hold the same value.

Added `CURLOPT_DEBUGFUNCTION` as a cURL option that accepts a `callable` that gets called during the request lifetime with the `CurlHandle` object, an integer containing the debug message type, and a string containing the debug message. The debug message type is one of the following constants: `CURLINFO_TEXT`, `CURLINFO_HEADER_IN`, `CURLINFO_HEADER_OUT`, `CURLINFO_DATA_IN`, `CURLINFO_DATA_OUT`,

`CURLINFO_SSL_DATA_IN`, `CURLINFO_SSL_DATA_OUT` Once this option is set, `CURLINFO_HEADER_OUT` must not be set because it uses the same libcurl functionality.

The `curl_getinfo` now returns an additional `posttransfer_time_us` key, containing the number of microseconds from the start until the last byte is sent. When a redirect is followed, the time from each request is added together. This value can also be retrieved by passing `CURLINFO_POSTTRANSFER_TIME_T` to the `curl_getinfo` `option` parameter. This requires libcurl 8.10.0 or later.

DOM

Added the Dom namespace with new classes as counterparts to the existing DOM classes (e.g. `Dom\Node` is the new `DOMNode`). These classes are compatible with HTML 5 and are WHATWG spec-compliant; solving long-standing bugs in the DOM extension. The old DOM classes remain available for backwards compatibility.

Added the DOMNode::compareDocumentPosition with its associated constants: `DOMNode::DOCUMENT_POSITION_DISCONNECTED`, `DOMNode::DOCUMENT_POSITION_PRECEDING`, `DOMNode::DOCUMENT_POSITION_FOLLOWING`, `DOMNode::DOCUMENT_POSITION_CONTAINS`, `DOMNode::DOCUMENT_POSITION_CONTAINED_BY`, `DOMNode::DOCUMENT_POSITION_IMPLEMENTATION_SPECIFIC`

It is now possible to pass any callable to DOMXPath::registerPhpFunctions. Furthermore, with DOMXPath::registerPhpFunctionNs, callbacks can now be registered that will use native function call syntax rather than using `php:function('name')`.

Intl

Added the `NumberFormatter::ROUND_HALFODD` to complement the existing `NumberFormatter::ROUND_HALFEVEN` functionality.

OpenSSL

Added support for Curve25519 + Curve448 based keys. Specifically x25519, ed25519, x448 and ed448 fields are supported in `openssl_pkey_new`, `openssl_pkey_get_details`, `openssl_sign`, and `openssl_verify` were extended to support those keys.

Implement PASSWORD_ARGON2 password hashing. Requires OpenSSL 3.2 and NTS build.

PCRE

The bundled pcre2lib has been updated to version 10.44. As a consequence, LoongArch JIT support has been added, spaces are now allowed between braces in Perl-compatible items, and variable-length lookbehind assertions are now supported.

With pcre2lib version 10.44, the maximum length of named capture groups has changed from `32` to `128`.

Added support for the `r` (PCRE2_EXTRA_CASELESS_RESTRICT) modifier, as well as the `(?r)` mode modifier. When enabled along with the case-insensitive modifier (`i`), the expression locks out mixing of ASCII and non-ASCII characters.

PDO

Added support for driver-specific subclasses in order to better support database-specific functionalities. The new classes are instantiatable either via calling the PDO::connect method or by instantiating one of the driver-specific subclasses directly.

Added support for driver specific SQL parsers. When a driver-specific parser is not available, the default parser is used. The default parser supports: single and double-quoted literals, with doubling as escaping mechanism, two-dashes and non-nested C-style comments

PDO_MYSQL

Added a custom parser supporting: single and double-quoted literals, with doubling and backslash as escaping mechanism, backtick literal identifiers and with doubling as escaping mechanism, two dashes followed by at least 1 whitespace, non-nested C-style comments, and hash-comments

PDO_PGSQL

Added a custom parser supporting: single and double-quoted literals, with doubling as escaping mechanism, C-style "escape" string literals (`E'string'`), dollar-quoted string literals, two-dashes and C-style comments (non-nested), support for `??` as escape sequence for the `?` operator

PDO_SQLITE

Added a custom parser supporting: single, double-quoted, and backtick literals, with doubling as escaping mechanism, square brackets quoting for identifiers, two-dashes and C-style comments (non-nested)

Phar

Added support for the Unix timestamp extension for Zip archives.

Readline

Added ability to change the `.php_history` path through the `PHP_HISTFILE` environment variable.

Reflection

`ReflectionAttribute` now contains a name property to improve the debugging experience.

ReflectionClassConstant::__toString and ReflectionProperty::__toString now returns the attached doc comments.

Multiple new methods and constants which are related to the lazy objects feature have been added: ReflectionClass::newLazyGhost, ReflectionClass::newLazyProxy, ReflectionClass::resetAsLazyGhost, ReflectionClass::resetAsLazyProxy, ReflectionClass::isUninitializedLazyObject, ReflectionClass::initializeLazyObject, ReflectionClass::markLazyObjectAsInitialized, ReflectionClass::getLazyInitializer, ReflectionProperty::skipLazyInitialization, ReflectionProperty::setRawValueWithoutLazyInitialization, `ReflectionClass::SKIP_INITIALIZATION_ON_SERIALIZE`, `ReflectionClass::SKIP_DESTRUCTOR`

SOAP

Added support for clark notation for namespaces in class map. It is now possible to specify entries in a class map with clark notation to resolve a type with a specific namespace to a specific class. For example: `'{http://example.com}foo' => 'FooClass'`.

Instances of DateTimeInterface that are passed to `xsd:datetime` or similar elements are now serialized as such instead of being serialized as an empty string.

Session persistence now works with a shared session module.

Standard

Added a new `RoundingMode` enum with clearer naming and improved discoverability compared to the `PHP_ROUND_*` constants. Moreover, four new rounding modes were added which are only available via the new `RoundingMode` enum.

XSL

It is now possible to use parameters that contain both single and double quotes.

It is now possible to pass any callable to XSLTProcessor::registerPhpFunctions.

Added XSLTProcessor::$maxTemplateDepth and XSLTProcessor::$maxTemplateVars to control the recursion depth of XSL template evaluation.

Zip

Added the `ZipArchive::ER_TRUNCATED_ZIP` constant, which was added in libzip 1.11.

□□□ #2
□ □□□ □□□ 2026-10-31 17:36:24 UTC
□ □□□ □□□ 2026-10-31 17:36:25 UTC