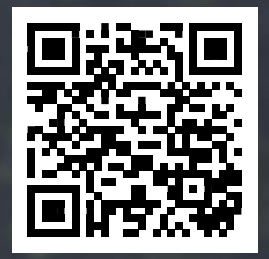
PHP 8.1 Enums

Ayesh Karunaratne | https://aye.sh/talk/midwest-php-2021-php-enums



PHP 8.1 Enums

Ayesh Karunaratne | https://aye.sh/talk/midwest-php-2021-php-enums

Ayesh Karunaratne

Freelance Software Developer, Security Researcher, Full-time traveler

Kandy, Sri Lanka - Everywhere

https://aye.sh | https://php.watch

@Ayeshlive | @phpwch

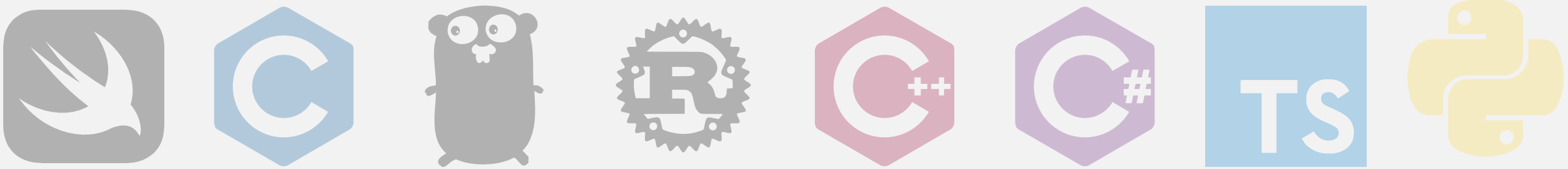
ayesh@aye.sh

PHP 8.1 Enums

PHP 8.1 Enumerations

PHP 8.1 Enumerations

PHP 8.1 Enumerations



Dec 04 2020	**Enumerations: RFC Created**
Feb 03 2021	**Voting started**
Feb 17 2021	**Voting ended: 44:7**
Apr 22 2021	**Midwest PHP 2021** 
Nov 25 2021	**PHP 8.1**

PHP 8.1: Enums

- Why we need Enums
- How Enums can help
- Enums in PHP 8.1
- Enum Semantics
- Usage Examples
- Trying out Enums today
- Backwards Compatibility

Why we need Enums

```php
$handle = curl_init();
$options = [
    CURLOPT_URL => 'https://example.com',
    CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_2_0,
    CURLOPT_RETURNTRANSFER => true,
];
curl_setopt_array($handle, $options);
curl_exec($handle);
```


```php
$handle = curl_init();
$options = [
    CURLOPT_URL => 'https://example.com',
    CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_2_0,
    CURLOPT_RETURNTRANSFER => true,
];
curl_setopt_array($handle, $options);
curl_exec($handle);
```


```php
$handle = curl_init();
$options = [
    CURLOPT_URL => 'https://example.com',
    CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_2_0,
    CURLOPT_RETURNTRANSFER => true,
];
var_dump($options);
curl_setopt_array($handle, $options);
curl_exec($handle);
```


```php
$handle = curl_init();
$options = [
    CURLOPT_URL => 'https://example.com',
    CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_2_0,
    CURLOPT_RETURNTRANSFER => true,
];
var_dump($options);
curl_setopt_array($handle, $options);
curl_exec($handle);
```

```
array(3) {
   [10002]=> string(19) "https://example.com"
   [84]=> int(3)
   [19913]=> bool(true)
}
```


```php
define ('CURLOPT_URL', 10002);
define ('CURLOPT_HTTP_VERSION', 84);
define ('CURL_HTTP_VERSION_1_1', 2);
define ('CURL_HTTP_VERSION_2_0', 3);
define ('CURLOPT_RETURNTRANSFER', 19913);
```

```php
$handle = curl_init();
$options = [
    CURLOPT_URL => 'https://example.com',
    CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_2_0,
    CURLOPT_RETURNTRANSFER => true,
];
var_dump($options);
curl_setopt_array($handle, $options);
curl_exec($handle);
```

```
array(3) {
   [10002]=> string(19) "https://example.com"
   [84]=> int(3)
   [19913]=> bool(true)
}
```


```php
define ('CURLOPT_URL', 10002);
define ('CURLOPT_HTTP_VERSION', 84);
define ('CURL_HTTP_VERSION_1_1', 2);
define ('CURL_HTTP_VERSION_2_0', 3);
define ('CURLOPT_RETURNTRANSFER', 19913);
```

```php
$handle = curl_init();
$options = [
    CURLOPT_URL => 'https://example.com',
    CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_2_0,
    CURLOPT_RETURNTRANSFER => true,
];
var_dump($options);
curl_setopt_array($handle, $options);
curl_exec($handle);
```

```
array(3) {
   [10002]=> string(19) "https://example.com"
   [84]=> int(3)
   [19913]=> bool(true)
}
```


```php
define ('CURLOPT_URL', 10002);
define ('CURLOPT_HTTP_VERSION', 84);
define ('CURL_HTTP_VERSION_1_1', 2);
define ('CURL_HTTP_VERSION_2_0', 3);
define ('CURLOPT_RETURNTRANSFER', 19913);
```

```php
$handle = curl_init();
$options = [
    CURLOPT_URL => 'https://example.com',
    CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_2_0,
    CURLOPT_RETURNTRANSFER => true,
];
var_dump($options);
curl_setopt_array($handle, $options);
curl_exec($handle);
```

```
array(3) {
   [10002]=> string(19) "https://example.com"
   [84]=> int(3)
   [19913]=> bool(true)
}
```


```php
function curl_setopt(CurlHandle $handle, int $option, mixed $value) : bool {}
```


```
function curl_setopt(CurlHandle $handle, int $option, mixed $value) : bool {}
```


```php
function curl_setopt(CurlHandle $handle, int $option, mixed $value) : bool {}
```

```php
curl_setopt($handle, 10002, 'https://example.com');
```

10002 - CURLOPT_URL

```php
curl_setopt($handle, 10003, 'https://example.com');
```

PHP Error: curl_setopt(): Argument #2 ($option) is not a valid cURL option in … on line …

```php
curl_setopt($handle, 10004, 'https://example.com');
```

10002 - CURLOPT_PROXY

```php
class Post {
    public const POST_STATUS_DRAFT = 'draft';
    public const POST_STATUS_PENDING = 'pending';
    public const POST_STATUS_RETURNED = 'returned';
    public const POST_STATUS_PUBLISHED = 'published';

    public string $status;

    public function updateStatus(string $status): void {}
}
```


```php
class Post {
    public const POST_STATUS_DRAFT = 'draft';
    public const POST_STATUS_PENDING = 'pending';
    public const POST_STATUS_RETURNED = 'returned';
    public const POST_STATUS_PUBLISHED = 'published';

    public string $status;

    public function updateStatus(string $status): void {}
}



$post = new Post();
$post->updateStatus(\Post::POST_STATUS_PUBLISHED);
```


```php
class Post {
    public const POST_STATUS_DRAFT = 'draft';
    public const POST_STATUS_PENDING = 'pending';
    public const POST_STATUS_RETURNED = 'returned';
    public const POST_STATUS_PUBLISHED = 'published';

    public string $status;

    public function updateStatus(string $status): void {}
}



$post = new Post();
$post->updateStatus('returned');
```


```php
class Post {
    public const POST_STATUS_DRAFT = 'draft';
    public const POST_STATUS_PENDING = 'pending';
    public const POST_STATUS_RETURNED = 'returned';
    public const POST_STATUS_PUBLISHED = 'published';

    public string $status;

    public function updateStatus(string $status): void {}
}



$post = new Post();
$post->updateStatus('returned');
```

```php
class Post {
    public const POST_STATUS_DRAFT = 'draft';
    public const POST_STATUS_PENDING = 'pending';
    public const POST_STATUS_RETURNED = 'returned';
    public const POST_STATUS_PUBLISHED = 'published';

    public string $status;

    public function updateStatus(string $status): void {}
}
```


```php
class Post {
    public const POST_STATUS_DRAFT = 'draft';
    public const POST_STATUS_PENDING = 'pending';
    public const POST_STATUS_RETURNED = 'returned';
    public const POST_STATUS_PUBLISHED = 'published';

    public string $status;

    public function updateStatus(string $status): void {
        if (  $status !== static::POST_STATUS_DRAFT
            && $status !== static::POST_STATUS_PENDING
            && $status !== static::POST_STATUS_RETURNED
            && $status !== static::POST_STATUS_PUBLISHED
        ) {
            throw new InvalidArgumentException('Invalid state');
        }
    }
}
```


How Enums Can Help

```
enum PostStatuses {

}
```


```
enum PostStatuses {
    case DRAFT;
    case PENDING;
    case RETURNED;
    case PUBLISHED;
}
```


```php
enum PostStatuses {
    case DRAFT;
    case PENDING;
    case RETURNED;
    case PUBLISHED;
}
```

```php
class Post {
    public const POST_STATUS_DRAFT = 'draft';
    public const POST_STATUS_PENDING = 'pending';
    public const POST_STATUS_RETURNED = 'returned';
    public const POST_STATUS_PUBLISHED = 'published';

    public string $status;

    public function updateStatus(string $status): void {
        if (  $status !== static::POST_STATUS_DRAFT
            && $status !== static::POST_STATUS_PENDING
            && $status !== static::POST_STATUS_RETURNED
            && $status !== static::POST_STATUS_PUBLISHED
        ) {
            throw new InvalidArgumentException('Invalid state');
        }
    }
}

$post = new Post();
$post->updateStatus(\Post::POST_STATUS_PUBLISHED);
```

```php
enum PostStatuses {
    case DRAFT;
    case PENDING;
    case RETURNED;
    case PUBLISHED;
}
```

```php
class Post {
    public const POST_STATUS_DRAFT = 'draft';
    public const POST_STATUS_PENDING = 'pending';
    public const POST_STATUS_RETURNED = 'returned';
    public const POST_STATUS_PUBLISHED = 'published';

    public string $status;

    public function updateStatus(string $status): void {
        if (  $status !== static::POST_STATUS_DRAFT
            && $status !== static::POST_STATUS_PENDING
            && $status !== static::POST_STATUS_RETURNED
            && $status !== static::POST_STATUS_PUBLISHED
        ) {
            throw new InvalidArgumentException('Invalid state');
        }
    }
}

$post = new Post();
$post->updateStatus(\Post::POST_STATUS_PUBLISHED);
```

```php
enum PostStatuses {
    case DRAFT;
    case PENDING;
    case RETURNED;
    case PUBLISHED;
}
```

```php
class Post {
    public const POST_STATUS_DRAFT = 'draft';
    public const POST_STATUS_PENDING = 'pending';
    public const POST_STATUS_RETURNED = 'returned';
    public const POST_STATUS_PUBLISHED = 'published';

    public string $status;

    public function updateStatus(string $status): void {
        if (  $status !== static::POST_STATUS_DRAFT
            && $status !== static::POST_STATUS_PENDING
            && $status !== static::POST_STATUS_RETURNED
            && $status !== static::POST_STATUS_PUBLISHED
        ) {
            throw new InvalidArgumentException('Invalid state');
        }
    }
}

$post = new Post();
$post->updateStatus(\Post::POST_STATUS_PUBLISHED);
```

```php
enum PostStatuses {
    case DRAFT;
    case PENDING;
    case RETURNED;
    case PUBLISHED;
}
```

```php
class Post {
    public const POST_STATUS_DRAFT = 'draft';
    public const POST_STATUS_PENDING = 'pending';
    public const POST_STATUS_RETURNED = 'returned';
    public const POST_STATUS_PUBLISHED = 'published';

    public PostStatuses $status;

    public function updateStatus(string $status): void {
        if (  $status !== static::POST_STATUS_DRAFT
            && $status !== static::POST_STATUS_PENDING
            && $status !== static::POST_STATUS_RETURNED
            && $status !== static::POST_STATUS_PUBLISHED
        ) {
            throw new InvalidArgumentException('Invalid state');
        }
    }
}

$post = new Post();
$post->updateStatus(\Post::POST_STATUS_PUBLISHED);
```

```php
enum PostStatuses {
    case DRAFT;
    case PENDING;
    case RETURNED;
    case PUBLISHED;
}
```

```php
class Post {
    public const POST_STATUS_DRAFT = 'draft';
    public const POST_STATUS_PENDING = 'pending';
    public const POST_STATUS_RETURNED = 'returned';
    public const POST_STATUS_PUBLISHED = 'published';

    public PostStatuses $status;

    public function updateStatus(PostStatuses $status): void {
        if (  $status !== static::POST_STATUS_DRAFT
            && $status !== static::POST_STATUS_PENDING
            && $status !== static::POST_STATUS_RETURNED
            && $status !== static::POST_STATUS_PUBLISHED
        ) {
            throw new InvalidArgumentException('Invalid state');
        }
    }
}

$post = new Post();
$post->updateStatus(\Post::POST_STATUS_PUBLISHED);
```

```php
enum PostStatuses {
    case DRAFT;
    case PENDING;
    case RETURNED;
    case PUBLISHED;
}
```

```php
class Post {
    public const POST_STATUS_DRAFT = 'draft';
    public const POST_STATUS_PENDING = 'pending';
    public const POST_STATUS_RETURNED = 'returned';
    public const POST_STATUS_PUBLISHED = 'published';

    public PostStatuses $status;

    public function updateStatus(PostStatuses $status): void {
        if (  $status !== static::POST_STATUS_DRAFT
            && $status !== static::POST_STATUS_PENDING
            && $status !== static::POST_STATUS_RETURNED
            && $status !== static::POST_STATUS_PUBLISHED
        ) {
            throw new InvalidArgumentException('Invalid state');
        }
    }
}

$post = new Post();
$post->updateStatus(\Post::POST_STATUS_PUBLISHED);
```

```php
enum PostStatuses {
    case DRAFT;
    case PENDING;
    case RETURNED;
    case PUBLISHED;
}
```

```php
class Post {
    public const POST_STATUS_DRAFT = 'draft';
    public const POST_STATUS_PENDING = 'pending';
    public const POST_STATUS_RETURNED = 'returned';
    public const POST_STATUS_PUBLISHED = 'published';

    public PostStatuses $status;

    public function updateStatus(PostStatuses $status): void {
        if (   $status !== static::POST_STATUS_DRAFT
            && $status !== static::POST_STATUS_PENDING
            && $status !== static::POST_STATUS_RETURNED
            && $status !== static::POST_STATUS_PUBLISHED
        ) {
            throw new InvalidArgumentException('Invalid state');
        }
    }
}

$post = new Post();
$post->updateStatus(\Post::POST_STATUS_PUBLISHED);
```

```php
enum PostStatuses {
    case DRAFT;
    case PENDING;
    case RETURNED;
    case PUBLISHED;
}
```

```php
class Post {
    public const POST_STATUS_DRAFT = 'draft';
    public const POST_STATUS_PENDING = 'pending';
    public const POST_STATUS_RETURNED = 'returned';
    public const POST_STATUS_PUBLISHED = 'published';

    public PostStatuses $status;

    public function updateStatus(PostStatuses $status): void {
        if (   $status !== static::POST_STATUS_DRAFT
            && $status !== static::POST_STATUS_PENDING
            && $status !== static::POST_STATUS_RETURNED
            && $status !== static::POST_STATUS_PUBLISHED
        ) {
            throw new InvalidArgumentException('Invalid state');
        }
    }
}

$post = new Post();
$post->updateStatus(\PostStatuses::PUBLISHED);
```

```php
enum PostStatuses {
    case DRAFT;
    case PENDING;
    case RETURNED;
    case PUBLISHED;
}
```

```php
class Post {
    public PostStatuses $status;

    public function updateStatus(PostStatuses $status): void {

    }
}

$post = new Post();
$post->updateStatus(\PostStatuses::PUBLISHED);
```


```php
enum PostStatuses {
    case DRAFT;
    case PENDING;
    case RETURNED;
    case PUBLISHED;
}
```

```php
function setIsSponsored(bool $sponsored): void {

}

function isSponsored(): bool {

}

setIsSponsored(true);
setIsSponsored(false);
```


Enums in PHP 8.1

Unit Enums

```
enum PostStatuses {
    case DRAFT;
    case PENDING;
    case RETURNED;
    case PUBLISHED;
}
```


Unit Enums

```php
enum PostStatuses implements UnitEnum {
    case DRAFT;
    case PENDING;
    case RETURNED;
    case PUBLISHED;
}
```

Unit Enums

```php
enum PostStatuses implements UnitEnum {
    case DRAFT;
    case PENDING;
    case RETURNED;
    case PUBLISHED;
}
```

```php
interface UnitEnum {
    public static function cases(): array;
}
```


Unit Enums

```php
enum PostStatuses implements UnitEnum {
    case DRAFT;
    case PENDING;
    case RETURNED;
    case PUBLISHED;
}
```

```php
interface UnitEnum {
    public static function cases(): array;
}
```

```php
echo PostStatuses::DRAFT->name;
// "DRAFT"
```


Backed Enums

Backed Enums extend Unit Enums

```php
enum PostStatuses: string {
    case DRAFT = 'draft';
    case PENDING = 'pending’;
    case RETURNED = 'returned';
    case PUBLISHED = 'published';
}
```


Backed Enums

Backed Enums extend Unit Enums

```php
enum PostStatuses: string implements BackedEnum {
    case DRAFT = 'draft';
    case PENDING = 'pending’;
    case RETURNED = 'returned';
    case PUBLISHED = 'published';
}
```


Backed Enums

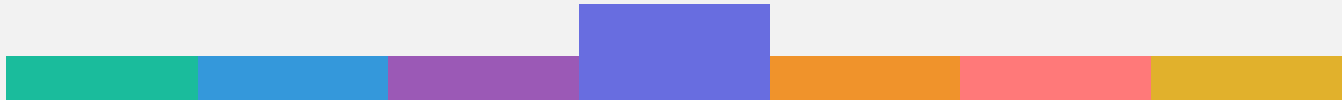
Backed Enums extend Unit Enums

```php
enum PostStatuses: string implements BackedEnum {
    case DRAFT = 'draft';
    case PENDING = 'pending’;
    case RETURNED = 'returned';
    case PUBLISHED = 'published';
}

interface BackedEnum extends UnitEnum {
    public static function from(int|string $value): static;
    public static function tryFrom(int|string $value): ?static;
}

echo PostStatuses::DRAFT->name;
// "DRAFT“

echo PostStatuses::DRAFT->value;
// "draft"
```


Enum Semantics

Enum

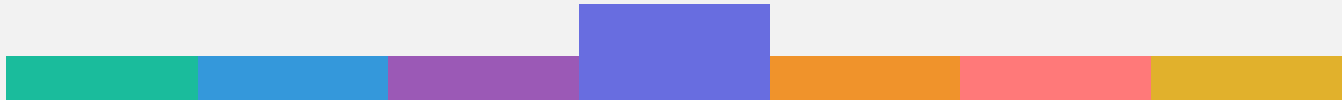
Enumerated type that contains a fixed number of members.



Enum

Enumerated **type** that contains a fixed number of members.

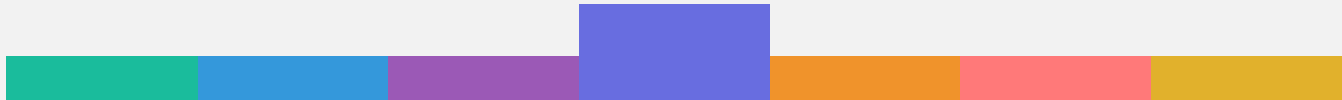
A type that is supported as parameter, return, and property type in PHP,
and the type is enforced by PHP itself.



Enum

Enumerated type that **contains** a fixed number of members.

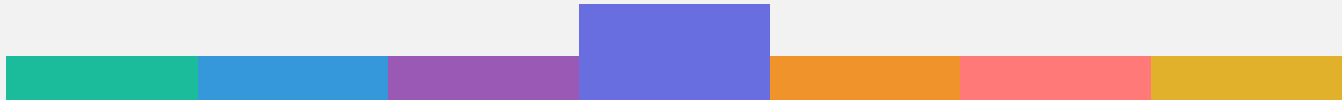
All members are contained within a declared Enum.



Enum

Enumerated type that contains a **fixed** number of members.

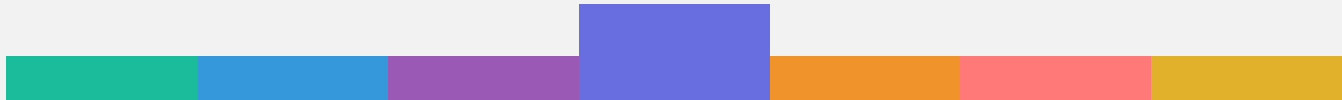
Members of an Enum is fixed at the declaration time.
An enumerated member is identical to the same member everywhere.
Enums must not contain state.



Enumerated types

```php
enum Suit {
    case Spades;
    case Hearts;
    case Clubs;
    case Diamonds;
}
```

```php
function play_card(Suit $suit, string $card) {}

function pick_a_suit(): Suit {
    return Suit::Spades;
}

play_card(Suit::Spades, 'A');
var_dump(pick_a_suit());
// enum(Suit::Spades)
```

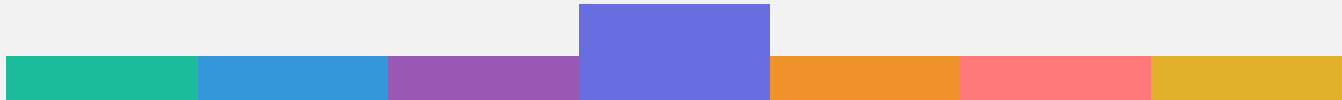
Enumerated types

```php
enum Suit {
    case Spades;
    case Hearts;
    case Clubs;
    case Diamonds;
}
```

```php
function play_card(Suit $suit, string $card) {}

function pick_a_suit(): Suit {
    return Suit::Spades;
}
```

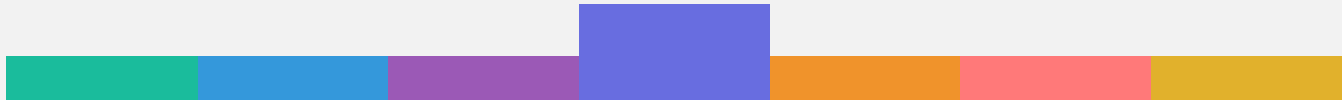
```php
play_card(Fruits::Apple);
play_card(Languages::English);
play_card('potato');
```

Fatal error: Uncaught TypeError: play_card(): Argument #1 ($suit) must be of type Suit, string given



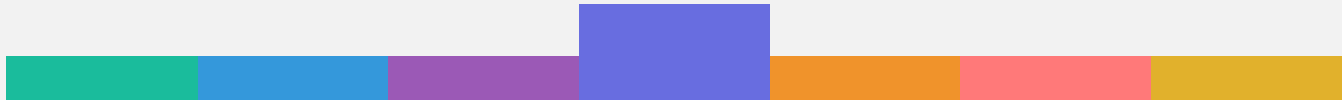
Closed Set

```
enum Suit {
    case Spades;
    case Hearts;
    case Clubs;
    case Diamonds;
}
```

Closed Set

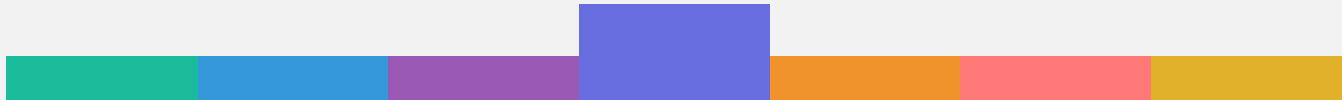
```
namespace Foo\Bar;

enum Suit {
    case Spades;
    case Hearts;
    case Clubs;
    case Diamonds;
}
```

Fixed Members

```
enum Suit {
    case Spades;
    case Hearts;
    case Clubs;
    case Diamonds;
}
```

```
Suit::Spades === Suit::Spades
```

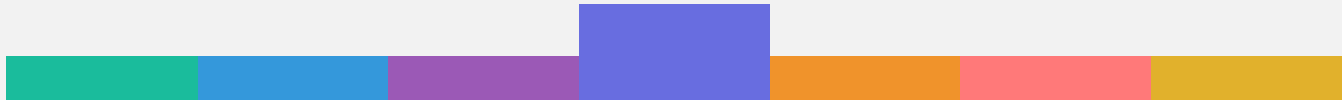



Fixed Members

```
enum Suit {
    case Spades;
    case Hearts;
    case Clubs;
    case Diamonds;
}
```

```
enum RussianSuit extends Suit {}
```

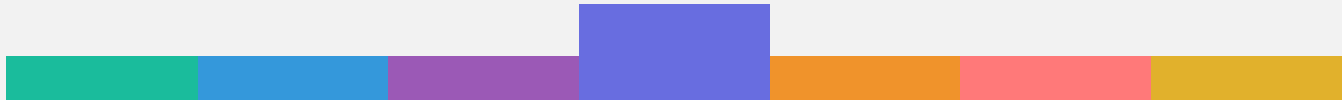
Parse error: syntax error, unexpected token "extends", expecting "{"



Fixed Members

```
enum Suit {
    case Spades;
    case Hearts;
    case Clubs;
    case Diamonds;

    private string $foo;
}
```

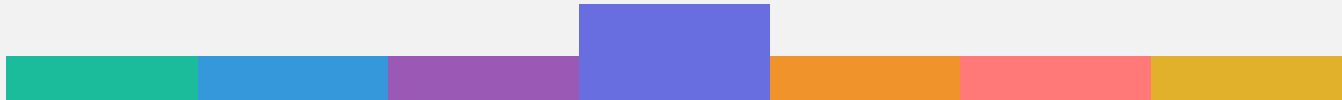
Fatal error: Enums may not include properties



Enums can have zero or more cases

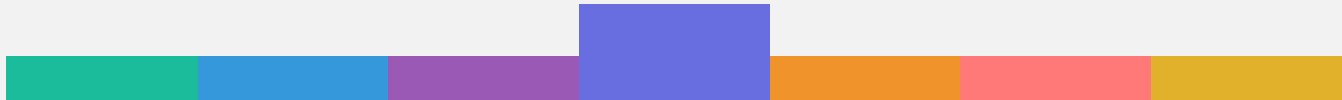
```
enum ErrorStates {
}
```

```
enum HTTPMethods {
        case GET;
        case POST;
}
```

Enums *may* have optional values

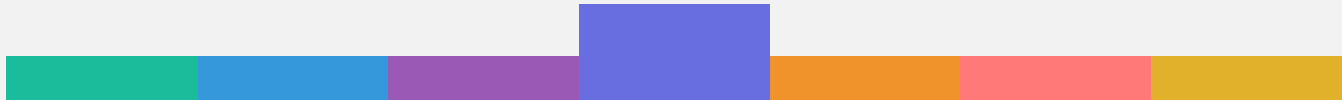
```
enum Suit: string {
    case Clubs = '♣';
    case Diamonds = '♦';
    case Hearts = '♥';
    case Spades = '♠';
}
```

Backed Enums *must* assign values for all cases

```
enum HTTPMethods: string {
    case GET;
    case POST;
}
```

Fatal error: Case GET of backed enum HTTPMethods must have a value



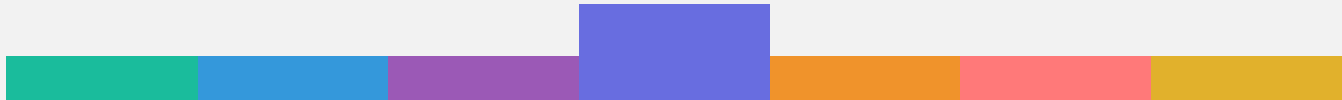
Enum cases and values *must* be unique

```php
enum Test {
    case FOO;
    case FOO;
}
```

Fatal error: Cannot redefine class constant Test::FOO

```php
enum Test: string {
    case FOO = 'baz';
    case BAR = 'baz';
}
```

Fatal error: Duplicate value in enum Test for cases FOO and BAR



Class Semantics

- Supports namespaces
- Supports traits
- Supports autoloading
- Supports magic constants
- Supports `instanceof`
- **Supports methods**

```php
namespace Foo\Bar;

enum PostStatuses: string implements EntityStatues {

    use TestTrait;

    case DRAFT = 'draft';
    case PENDING = 'pending';
    case RETURNED = 'returned';
    case PUBLISHED = 'published’;

    public static function showOff(): void {
        echo __CLASS__ . static::class;
    }

}
```

Usage Examples

```
enum PostStatuses: string {
    case DRAFT = 'draft';
    case PENDING = 'pending’;
    case RETURNED = 'returned';
    case PUBLISHED = 'published';
}
```


```php
enum PostStatuses: string {
    case DRAFT = 'draft';
    case PENDING = 'pending’;
    case RETURNED = 'returned';
    case PUBLISHED = 'published';
}
```

```php
class Post {
    private int $id;
    private string $title;
    private PostStatuses $status;

    public function __construct(
        int $id,
        string $title
    ) {
        // ...
    }
}
```

Usage Examples

```php
enum PostStatuses: string {
    case DRAFT = 'draft';
    case PENDING = 'pending’;
    case RETURNED = 'returned';
    case PUBLISHED = 'published';
}
```

```php
class Post {
    private int $id;
    private string $title;
    private PostStatuses $status;

    public function __construct(
        int $id,
        string $title
    ) {
        // ...
    }
}
```

Usage Examples

```php
enum PostStatuses: string {
    case DRAFT = 'draft';
    case PENDING = 'pending’;
    case RETURNED = 'returned';
    case PUBLISHED = 'published';
}
```

```php
class Post {
    private int $id;
    private string $title;
    private PostStatuses $status;

    public function __construct(
        int $id,
        string $title
    ) {
        // ...
    }

    public function updateStatus(PostStatuses $status): void {
        $this->status = $status;
    }

    public function getStatus(): PostStatuses {
        return $this->status;
    }
}
```

Usage Examples

```php
enum PostStatuses: string {
    case DRAFT = 'draft';
    case PENDING = 'pending’;
    case RETURNED = 'returned';
    case PUBLISHED = 'published';
}
```

```php
class Post {
    private int $id;
    private string $title;
    private PostStatuses $status;

    public function __construct(
        int $id,
        string $title
    ) {
        // ...
    }

    public function updateStatus(PostStatuses $status): void {
        $this->status = $status;
    }

    public function getStatus(): PostStatuses {
        return $this->status;
    }
}
```

```php
$stmt = $pdo->prepare("
    SELECT *
    FROM posts
    WHERE post_status=?");
$stmt->execute([
    PostStatuses::PUBLISHED->value
]);
$post = $stmt->fetch();
```

Usage Examples

```php
enum PostStatuses: string {
    case DRAFT = 'draft';
    case PENDING = 'pending’;
    case RETURNED = 'returned';
    case PUBLISHED = 'published';
}
```

```php
class Post {
    private int $id;
    private string $title;
    private PostStatuses $status;

    public function __construct(
        int $id,
        string $title
    ) {
        // ...
    }

    public function updateStatus(PostStatuses $status): void {
        $this->status = $status;
    }

    public function getStatus(): PostStatuses {
        return $this->status;
    }
}
```

```php
$stmt = $pdo->prepare("
    SELECT *
    FROM posts
    WHERE post_status=?");
$stmt->execute([
    PostStatuses::PUBLISHED->value
]);
$post = $stmt->fetch();
```

Usage Examples

```php
enum PostStatuses: string {
    case DRAFT = 'draft';
    case PENDING = 'pending’;
    case RETURNED = 'returned';
    case PUBLISHED = 'published';
}
```

```php
class Post {
    private int $id;
    private string $title;
    private PostStatuses $status;

    public function __construct(
        int $id,
        string $title
    ) {
        // ...
    }

    public function updateStatus(PostStatuses $status): void {
        $this->status = $status;
    }

    public function getStatus(): PostStatuses {
        return $this->status;
    }
}
```

```php
$sql = "
    INSERT INTO
        posts (id, title, post_status)
     VALUES
         (:id, :title, :post_status)";
$stmt= $pdo->prepare($sql);
$stmt->execute([
    'id' => $post->getId(),
    'title' => $post->getTitle(),
    'post_status' => $post->getStatus()->value,
]);
```

Usage Examples

```php
enum PostStatuses: string {
    case DRAFT = 'draft';
    case PENDING = 'pending’;
    case RETURNED = 'returned';
    case PUBLISHED = 'published';
}
```

```php
class Post {
    private int $id;
    private string $title;
    private PostStatuses $status;

    public function __construct(
        int $id,
        string $title
    ) {
        // ...
    }

    public function updateStatus(PostStatuses $status): void {
        $this->status = $status;
    }

    public function getStatus(): PostStatuses {
        return $this->status;
    }
}
```

```php
$sql = "
    INSERT INTO
        posts (id, title, post_status)
     VALUES
         (:id, :title, :post_status)";
$stmt= $pdo->prepare($sql);
$stmt->execute([
    'id' => $post->getId(),
    'title' => $post->getTitle(),
    'post_status' => $post->getStatus()->value,
]);
```

Usage Examples

```php
enum PostStatuses: string {
    case DRAFT = 'draft';
    case PENDING = 'pending’;
    case RETURNED = 'returned';
    case PUBLISHED = 'published';
}
```

```php
class Post {
    private int $id;
    private string $title;
    private PostStatuses $status;

    public function __construct(
        int $id,
        string $title
    ) {
        // ...
    }

    public function updateStatus(PostStatuses $status): void {
        $this->status = $status;
    }

    public function getStatus(): PostStatuses {
        return $this->status;
    }
}
```

```php
$result = [
    'id' => 42,
    'title' => 'PHP Enums',
    'post_status' => 'published',
];

$post = new Post(
    $result['id'],
    $result['title']
);

$post->updateStatus(
    PostStatuses::from($result['post_status'])
);
```

Usage Examples

```php
enum PostStatuses: string {
    case DRAFT = 'draft';
    case PENDING = 'pending’;
    case RETURNED = 'returned';
    case PUBLISHED = 'published';
}
```

```php
class Post {
    private int $id;
    private string $title;
    private PostStatuses $status;

    public function __construct(
        int $id,
        string $title
    ) {
        // ...
    }

    public function updateStatus(PostStatuses $status): void {
        $this->status = $status;
    }

    public function getStatus(): PostStatuses {
        return $this->status;
    }
}
```

```php
$result = [
    'id' => 42,
    'title' => 'PHP Enums',
    'post_status' => 'published',
];

$post = new Post(
    $result['id'],
    $result['title']
);

$post->updateStatus(
    PostStatuses::from($result['post_status'])
);
```

Usage Examples

Trying out Enums today

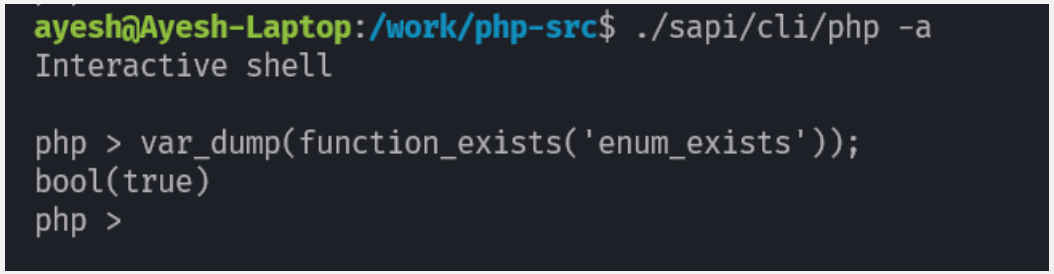
Try it online with 3v4l.org

Nightly Docker Images

```
docker pull phpdaily/php:8.1-dev
```


Self-compile PHP from source

```
$  git clone git@github.com:php/php-src.git
$  ./buildconf
$  ./configure
$  make -j$(nproc)
$  ./sapi/cli/php -a
```



Backwards Compatibility

Enums is a new syntax

Enums is a new syntax introduced in PHP 8.1, and not supported in older PHP versions.

Parse error: syntax error, unexpected identifier "PostStatuses"

User-land PHP implementations

https://github.com/myclabs/php-enum

```php
use MyCLabs\Enum\Enum;

class PostStatuses extends Enum {
    private const DRAFT = 'draft';
    private const PENDING = 'pending';
    private const RETURNED = 'returned';
    private const PUBLISHED = 'published';
}
```

Further Resources

- https://php.watch/versions/8.1/enums
- https://php.watch/versions/8.1
- https://wiki.php.net/rfc/enumerations
- https://phpinternals.news/73
- https://github.com/php/php-src/pull/6489/
- https://externals.io/message/112626
- https://github.com/phpdaily/php
- https://3v4l.org/

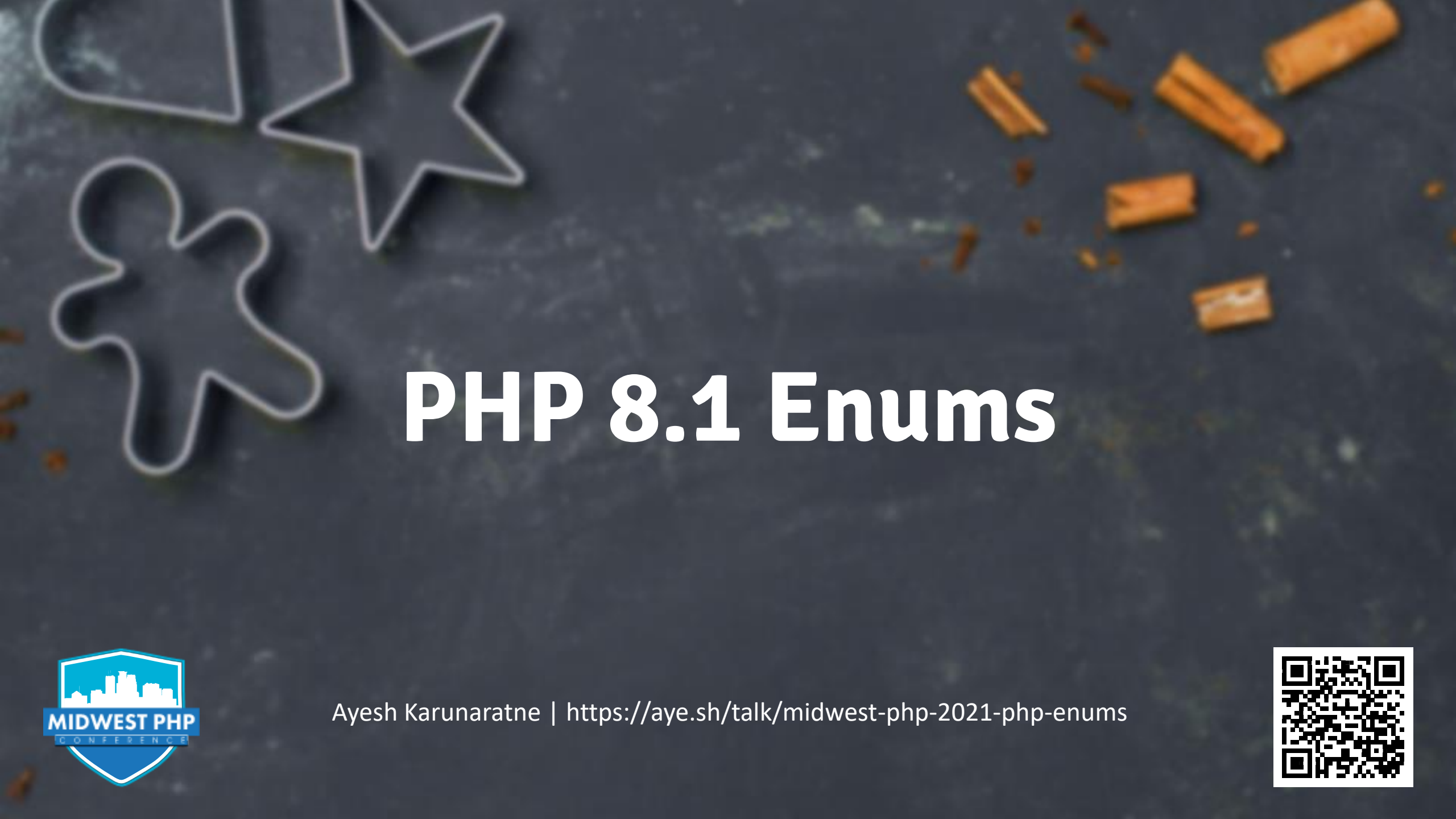
Questions?

No question is too small.

@Ayeshlive ayesh@php.watch

https://aye.sh/talk/midwest-php-2021-php-enums


dziękuję
Ďakujem
tak
arigatô
paldies
mahalo
kop khun
děkuji
diolch
dankie
köszönöm
хвала
shukran
cảm ơn bạn
tänan
a dank
gràcies
ngiyabonga
Баярлалаа
dhanyavād
Дякую
Благодарам
ευχαριστώ
спасибо
благодаря
tack
grazie
Mh’gōi
Dank u
Благодаря ти
gracias
ස්තුතියි
mulţumesc
ačiū
תודה.
nandri
takk
danke
faleminderit
Xièxiè
choukrane
teşekkür ederim
obrigado
kiitos
Շնորհակալություն
terima kasih
hvala
grazzi
THANK YOU


PHP 8.1 Enums
MIDWEST PHP
CONFERENCE
Ayesh Karunaratne | https://aye.sh/talk/midwest-php-2021-php-enums