PHP Meetup

Welcome

sourcetoad

PHP7.x

Type Declarations + more

sourcetoad

PHP7: December 3, 2015

- 2018 already! You **need** to be on PHP7+
  - Security enhancements
  - Speed enhancements
  - Features and deprecations
  - more and more!

PHP5: Type Hints

```php
// array
function(array $test) {}

// callable
function(callable $test) {}

// class/interface
class Cat {}
function(Cat $test) {}
```

PHP5: Consistency is key

- function foo (array $test) {}
- foo(“”);
- Argument 1 passed to foo() must be of the type array, string given

PHP5: Why do this?

- IDE Benefits (Auto complete)
- Catch mistakes earlier
- Sort of a “Contract”, I expect this
- Interoperability
- Self-Descriptive

So what did **PHP7** add?

- **Scalar** type declarations
  - string, int, float, boolean
- **Return type** declarations
- **Strict** directive
  - declare(strict_types = 1);

Scalar type hints!

```php
// Sum
function sum(int $a, int $b) {
  return $a + $b;
}
```

PHP7: Scalar Typehint

- $a and $b must be an int
- You **know** $a/$b in functions are ints now.
- What are we returning?
  - **Depends on logic.**

What about **returns?**

```php
// Sum
function sum(int $a, int $b): int {
    return $a + $b;
}
```

PHP7: Typehint Return

- Yay return is an int.
- Logic must abide by that.
- No returning an array/string
- Like docblocks, but enforced at code

What if I made a **mistake**

```php
// Sum
function sum(int $a, int $b) {
  return $a . $b;
}
```

```php
<?php

function sum(int $a, int $b): int {
  return $a . $b;
}
// sum(2, 3);
// 23
// wat
```

```php
<?php
declare(strict_types = 1);

function sum(int $a, int $b): int {
  return $a . $b;
}
// sum(2, 3);
// Uncaught TypeError: Return value of sum()
// must be of the type integer, string returned
```

```php
<?php
declare(strict_types = 0);

function sum(int $a, int $b): int {
   return $a + $b;
}
// sum(2.3, 3);
// 5
```

Letting PHP Guess **sucks**.

- 2.3 casted as int, is 2
- Intended? Intentional? Who knows.

```php
<?php
declare(strict_types = 1);

function sum(int $a, int $b): int {
  return $a + $b;
}
// sum(2.3, 3);
// Uncaught TypeError: Argument 1 passed to
```

No guessing. **Strict**

- Personal Preference?
  - Some prefer loose typing
- What about frameworks?
  - Yii2, Laravel, etc?
  - This is a **breaking** change

Bonus: Null Coalescing Operator

- ??
- ^ Thats the operator
- $_GET[‘foo’] ?? ‘unknown’;
- isset($_GET[‘foo’]) ? $_GET[‘user’] : ‘unknown’;

```php
<?php
declare(strict_types = 1);

function sum(int $a, int $b): int {
    if ($a == 0 || $b == 0) {
        return null;
    }
    return $a + $b;
}
// sum(0, 5) = Uncaught TypeError
```

Null and a data type?

- Hello **PHP7.1**
- Meet the ? operator
- This or **null**

```php
<?php
declare(strict_types = 1);

function sum(int $a, int $b): ?int {
    if ($a == 0 || $b == 0) {
        return null;
    }
    return $a + $b;
}
// sum(0, 5) = null
```

Can you return **nothing** (void)?

- Once again **PHP7.1** to the rescue
- **void** return type added. Joins the list:
  - scalars
  - arrays
  - classes / interfaces
  - callables

Is an **object** a scalar?

- Well an object is usually a class
- Can you just return **object**?
- **PHP7.2** fixes this.
- **object** joins the type group

Bonus: Anonymous classes

- Much like anonymous functions
- **PHP7** brings them to classes!
- Use in tests or mocking

```php
<?php
declare(strict_types = 1);

$test->setClass(new class {
  public function something(int $a): int {
    return $a;
  }
});
```

Bonus: PHP72 - **Type Widening**

- Why doesn’t Laravel/Yii2 use type hinting?
- Imagine every class/plugin/etc built off
- Massive breaking changes, until PHP72
- Widen child classes as to not break code

```php
<?php

class Cat {
  public function breed(Cat $cat) { };
}

class Lion {
  public function breed($cat) {};
}
```

Bonus: PHP72 - **Argon2**

- Hashing evolves, **Bcrypt** is not the best
- Memory, time and parallelism cost
- Very easy to use!
- You should be **scared** if you are still using MD5

```php
<?php

password_hash('password', PASSWORD_ARGON2I);

// $argon2i$v=19$m=1024,t=2,p=2$TVNXSTQwOFVoUVVl
```

Bonus: PHP72 - **Libsodium**

- Crypto is hard
- PHP was outdated.
- Encryption, Decryption, Signatures, etc
- Libmcrypt is old (2007) :/

```php
<?php

// key, secret message
$secret = sodium_crypto_secretbox_keygen();
$message = 'Sensitive information';

// nonce and then encrypte text
$nonce = random_bytes(SODIUM_CRYPTO_SECRETBOX_NONCEBYTES);
$encrypted = sodium_crypto_secretbox($message, $nonce, $secret_key);

// decryption
$decrypted = sodium_crypto_secretbox_open($encrypted, $nonce, $secret);
```

PHP7+ has some goodies.

- Push for upgrades
- Move to the future!
- Deprecations are not scary

@iBotPeaches

// +ConnorTumbleson

sourcetoad