

# Code with confidence using PHPStan

1. What does code confidence mean to me
2. What is static analysis
3. How do we install/run/configure PHPStan
4. How to increase code confidence using PHPStan

# \$ whoami = Peter Fisher

- PHP Contractor from the UK
- Playing with PHP > 20 years
- Host of the How To Code Well
  - Podcast [howtocodewell.fm](https://howtocodewell.fm)
  - YouTube channel [youtube.com/howtocodewell](https://youtube.com/howtocodewell)
  - Twitch live coders team [howtocodewell.net/live](https://howtocodewell.net/live)
  - Tutorials and courses [howtocodewell.net](https://howtocodewell.net)

# Get the slides

<http://peterfisher.me.uk/slides/code-with-confidence-using-phpstan.html>

# #1

## **What does code confidence mean to me**

**There are three types of projects  
that every programmer deals with  
during their career**

# 1# New projects

# 2# Legacy projects



# 3# Migrations/rebuilds

# The dream

# New projects

Start clean, continue clean whilst building up confidence  
with the code

# Legacy projects

Quickly identify issues whilst building up confidence with the code

# Migrated projects

Ensure the migration is smooth with as little disruption as possible

# Code confidence

The dreaded 3am phone call/Slack message on Saturday  
after the Friday production deployment

VS

Knowing your code and everyone else's code will get  
checked before it goes anywhere near production

# How do we get there

# Add Static Analysis to your toolbox



# #2

## What is Static Analysis

# From Wikipedia

"Static program analysis is the analysis of computer software performed without executing any programs, in contrast with dynamic analysis, which is performed on programs during their execution"

# What does that mean?

- Static analysis will search code for non coding compliance without the need for code execution.
- It compares the code against a given set of rules
- It tells you which file and line doesn't conform to which rule
- It prevents very bad things from happening

# What's the point?

## PHP type system is at runtime

**A bug found at runtime will  
always cost more than a bug  
found during static analysis.**

# Type checking

```
$var = new stdClass() + 5;  
echo $var;
```

```
// PHP Warning:  Uncaught TypeError: Unsupported operand types
```

# But my code works?

- It could be risky
- It could be broken but working
- It may not be future proof

# #3

## PHPStan has entered the chat

- [phpstan.org](https://phpstan.org)
- Is free and open source
- Has pro paid features



# How to install

```
$ composer require --dev phpstan/phpstan
```

# Your first run

```
$ ./vendor/bin/phpstan analyse src
```

# When things go well

```
root@768e64cf6e00:/var/www/html# ./vendor/bin/phpstan analyse src
```

```
293/293 [████████████████████████████████████████████████████████████████████████████████] 100%
```

```
[OK] No errors
```

# Catching errors

```
root@768e64cf6e00:/var/www/html# ./vendor/bin/phpstan analyse src
293/293 [████████████████████████████████████████] 100%
```

Line	Downloader/CodeDownloader.php
84	Method App\Downloader\CodeDownloader::getFilename() should <b>return</b> string but returns string null.

```
[ERROR] Found 1 error
```

# The fix

```
public function getFilename(): string
{
    return $this->course?->getCode()?->getFileName();
}
```

```
public function getFilename(): ?string
{
    return $this->course?->getCode()?->getFileName();
}
```

# Run levels

- There are 10 run levels (0-9) that change the strictness of the checks.
- Level 0 is used by default.
- Running level 5 will run all the levels from 0-5

# How to run PHPStan at a given level

```
./vendor/bin/phpstan analyse -l 5 src
```

# How to ignore code

```
private $firstName /** @phpstan-ignore-line */  
  
/** @phpstan-ignore-next-line */  
private $lastName
```



# How to configure

- Neon format (phpstan.neon, phpstan.neon.dist)
- CLI

# Neon format is similar to YAML

```
parameters:  
  level: 6  
  paths:  
    - src  
    - tests
```

# Priority order

1. If a config file is supplied via CLI then it will be used  
( `-c` )
2. Otherwise, if `phpstan.neon` exists then it will be used
3. Otherwise, if `phpstan.neon.dist` exists that it will be used
4. If no config is supplied then defaults will be used

# Git

- Put `phpstan.neon.dist` in source control
- Let devs create their own `phpstan.neon`
- Add `phpstan.neon` to `.gitignore`

# Including config files

```
includes:
```

- `phpstan.neon.dist`
- `phpstan_test.neon.dist`

# Checking paths

```
parameters:  
  paths:  
    - src  
    - tests
```

```
./vendor/bin/phpstan analyse src tests
```

# Excluding files

```
parameters:  
  excludePaths:  
    - tests/*/data/*
```

# Ignoring errors

```
parameters:  
  ignoreErrors:  
    - '#Function pcntl_open not found\.#'
```



# Lots more config

See <https://phpstan.org/config-reference> for more

# #4

## How to increase code confidence using PHPStan

# Recommendations for any project

# Test order is important

PHPCs -> PHPStan -> PHPUnit

# One command to rule them all

```
$ make tests
```

```
$ composer test
```

# Use a CI

# Only test your code

# Be careful with upgrades



# Use other extensions that match your setup

`phpstan/phpstan-doctrine`

`phpstan/phpstan-symfony`

# Recommendations for new projects

# Run at max level

```
./vendor/bin/phpstan analyse -l max src
```

```
parameters:  
  level: max  
  paths:  
    - src
```

# Get stricter

<https://github.com/phpstan/phpstan-strict-rules>

```
composer require --dev phpstan/phpstan-strict-rules
```

```
includes:
```

```
– vendor/phpstan/phpstan-strict-rules/rules.neon
```

OR

<https://github.com/phpstan/extension-installer>

# Recommendations for legacy projects

# Run the highest level once

# Start small and go gradually

**Make sure you have tests to  
back up your changes**



# 3 Confidence levels for legacy projects

**1) PHPStan is already in use  
and is running at the highest  
level and working well**

- High confidence level

## 2) PHPStan is installed but using a low run level

- Low confidence level

How do you upgrade PHPStan on a legacy project?

# 3) PHPStan is not installed

- Very low confidence level

How do you install PHPStan on a legacy project?

1. Get the by in of the team
2. Run at the highest level
3. Generate a baseline level
4. Put the fixes in a separate branch/pr
5. Rinse and repeat

# Generate a Baseline level

```
vendor/bin/phpstan analyse --level 7 \  
--configuration phpstan.neon \  
src/ tests/ --generate-baseline
```

includes:

- phpstan-baseline.neon

# Generics are Awesome

Loop over an array of products getting the ID of each  
product

Sounds easy right?

# Oh no

```
$products = [  
    new PreOrder(),  
    new Subscription(),  
    new Product(),  
    'SKUABCD',  
];
```



# A work around

```
foreach ($products as $product) {  
    if (!$product instanceof Product ||  
        !$product instanceof Subscription ||  
        !$product instanceof PreOrder ||  
    )  
    {  
        continue;  
    }  
  
    $id = $product->getId();  
    //..  
}
```

```
function getProductIds(array $products) {  
    foreach ($products as $product) {  
        // Is $product actually an instance of Product?  
    }  
}
```

# Messy code

- Checks get out of hand
- Not very readable
- Prone to mistakes

```
/**
 * @param array<int, Product|Subscription|PreOrder|string> $products
 * @return array<int, int>
 */
function getProductIds(array $products): array
{
    $ids = [];
    foreach($products as $product){
        if(is_string($product)){
            continue;
        }

        $ids[] = $product->getId()
    }
    return $ids;
}
```

# When to use annotations or native type hints

- It's up to you!
- Don't double up
- Use native type hints where possible
- Use annotations when you can't use native type hints

```
/**  
 * @return array<string, int>  
 */  
function getItems(): array  
{  
    return [  
        'hello' => 1,  
        'world' => 2  
    ];  
}
```



# Static Analysis could save you money

If you're relying on Bugsnag or Sentry to catch errors that  
Static Analysis can catch then you're doing it wrong



# Thank you

@pfwd