Our Standards And Why We Use Them

Developer at Jump 24 - Working with Laravel and CakePHP using TDD

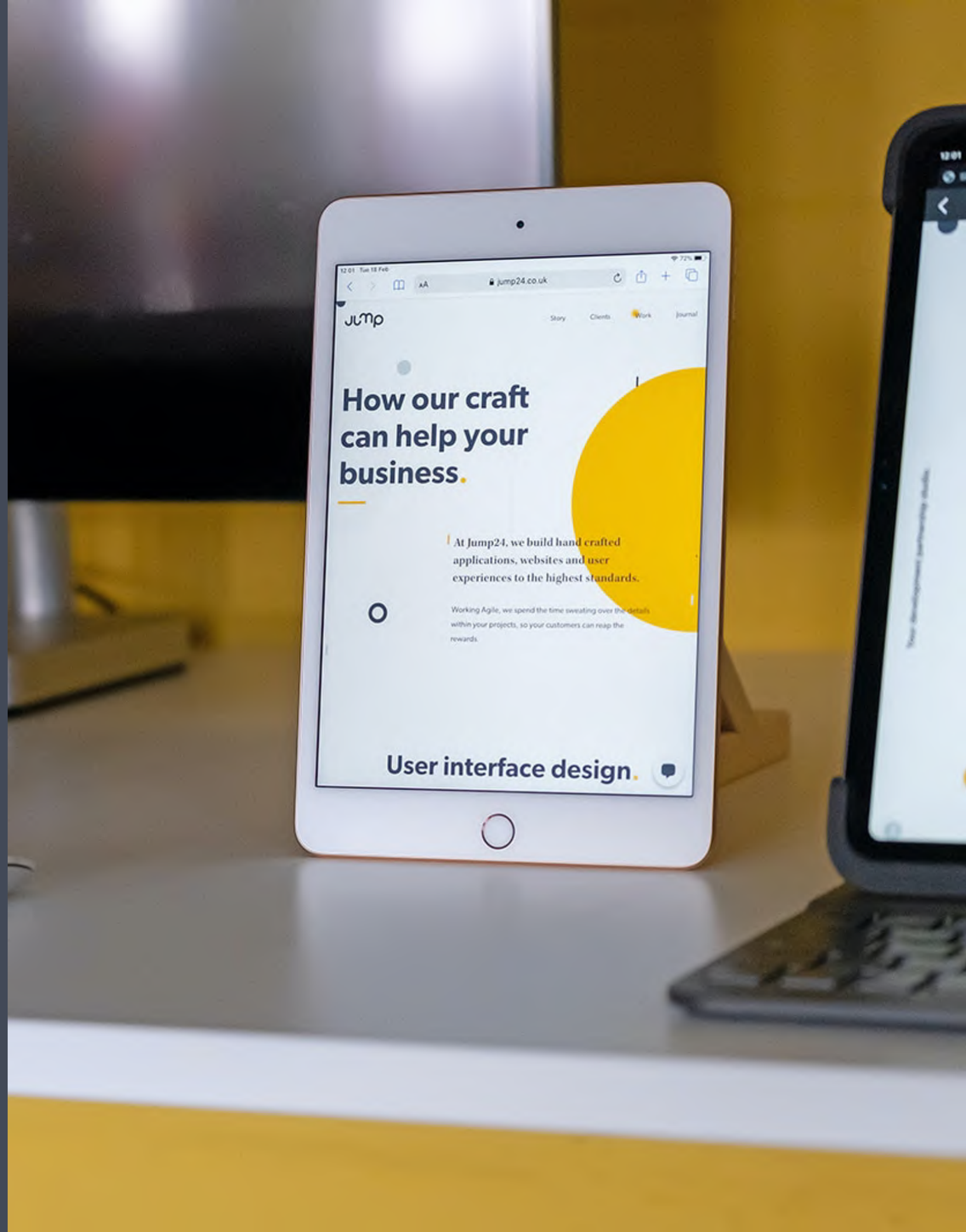
WHO AM I? WHAT AM I DOING HERE?

Twitter: ccmiller2018

Email: chris@jump24.co.uk

Website: jump24.co.uk

WE ARE RECRUITING ACROSS THE BOARD - REMOTE POSITIONS, MUST BE UK RESIDENT



Why Do We Need Standards?

WE HAVE A LOT OF SYSTEMS WE USE!

What Is Our Stack?

- CakePHP
- Terraform
- AWS
- PHP
- Nginx
- Laravel
- Inertia
- React
- Sass
- MySQL
- Redis
- HTML5
- CSS3
- Javascript
- Webpack
- Docker

Key Takeaway

STANDARDS HELP - THEY ENABLE DEVELOPERS TO WORK TOGETHER BETTER - SIMPLIFIES CODE REVIEW - KEEPS CODE CONSISTENT ACROSS PROJECTS

PHP

THE STANDARDS WE APPLY

PHP Standards Are Our Centrepiece

- Linter
- ECS
- PHPStan
- Tests

Everything we do is thought in terms of the "hit by a bus" scenario

If a developer were not available tomorrow, could someone else pick up their work?

Linter

NOT JUST MAKING IT PRETTY

A Bad Code Example

```php
<?php

const MY_MESSAGE = 'something'

function echoMe() {
    echo (MY_MESSAGE)
    echo ($otherMessage);
}
```

At First Glance This Looks Ok

- But there are errors here
- Quite a few of them

```
→  linter php linter.php
PHP Parse error:  syntax error, unexpected 'function' (T_FUNCTION), expecting ',' or ';' in /Users/ccmiller/Talks/standards/code-examples/linter/linter.php on line 5

Parse error: syntax error, unexpected 'function' (T_FUNCTION), expecting ',' or ';' in /Users/ccmiller/Talks/standards/code-examples/linter/linter.php on line 5
→  linter
```

Our Code Fails To Execute

Bringing In Some Packages

```
"require-dev": {
    "php-parallel-lint/php-console-highlighter": "^0.5.0",
    "php-parallel-lint/php-parallel-lint": "^1.3"
},
"scripts": {
    "lint": [
        "vendor/bin/parallel-lint ./ --blame --exclude vendor"
    ]
}
```

Two Packages And A Script

- Run Composer Install
- Run Composer Lint

```
➜  linter composer lint
> vendor/bin/parallel-lint ./ --blame --colors --exclude vendor
PHP 7.2.34 | 10 parallel jobs
X                                                                   1/1 (100 %)



Checked 1 files in 0 seconds
Syntax error found in 1 file


---------------------------------------------------------------
Parse error: ./linter.php:5
    3| const MY_MESSAGE = 'something'
    4|
  > 5| function echoMe() {
    6|     echo (MY_MESSAGE)
    7|     echo ($otherMessage);
Unexpected 'function' (T_FUNCTION), expecting ',' or ';' in ./linter.php on line 5
Script vendor/bin/parallel-lint ./ --blame --colors --exclude vendor handling the lint event returned with error code 1
➜  linter
```

Now We Can See Whats Wrong

Our Code Fixed

```php
<?php

const MY_MESSAGE = 'something';

function echoMe(string $otherMessage)
{
    echo (MY_MESSAGE);
    echo ($otherMessage);
}

echoMe('else');
```

Our Code Executes

Use Our Standards Packages

NOW WE START TO ENFORCE OUR PARTICULAR CODING STYLE

Bringing In All The Packages We Need

```
"require-dev": {
    "jumptwentyfour/php-coding-standards": "^0.0.1",
    "jumptwentyfour/project-analysers": "^0.0.2",
    "php-parallel-lint/php-console-highlighter": "^0.5.0",
    "php-parallel-lint/php-parallel-lint": "^1.3",
},
```

And The Scripts

```
"scripts": {
    "lint": [
        "vendor/bin/parallel-lint ./ --blame --exclude vendor"
    ],
    "ecs": [
        "vendor/bin/ecs check"
    ],
    "static": [
        "vendor/bin/phpstan analyse --memory-limit=2G"
    ],
    "test": [
        "vendor/bin/pest --parallel"
    ],
}
```

Now A PHPStan.neon file

```
includes:
    - ./vendor/jumptwentyfour/php-coding-standards/phpstan.neon
    - ./phpstan-baseline.neon
parameters:

    # The level 8 is the highest level
    level: 8

    excludePaths:
        - 'vendor/*'
        - '_ide_helper.php'
        - '_ide_helper_models.php'
        - '.phpstorm.meta.php'

    checkMissingIterableValueType: false
```

AND DONT FORGET A BLANK PHPSTAN-BASELINE.NEON FILE

And ECS.php

```php
<?php

use Symfony\Component\DependencyInjection\Loader\Configurator\ContainerConfigurator;
use Symplify\EasyCodingStandard\ValueObject\Option;

return static function (ContainerConfigurator $containerConfigurator): void {
    $containerConfigurator->import(__DIR__ . '/vendor/jumptwentyfour/php-coding-standards/ecs.php');

    $parameters = $containerConfigurator->parameters();

    $parameters->set(Option::PATHS, [
        __DIR__ . '/app',
        __DIR__ . '/tests',
    ]);
};
```

So What Are We Using Here?

LINTING, STATIC ANALYSIS, CODING STANDARDS

Why Linting?

IT HIGHLIGHTS CODE ERRORS

Why ECS?

IT ALLOWS US TO ENSURE OUR STANDARDS FOR CODE ARE MET, EXTENDING THIS EASILY, AND ALLOWS FOR IN-EDITOR HIGHLIGHTING OF OUR CUSTOM STANDARDS

What Standards Do We Apply?

PSR-12 AND RELATED OTHER STANDARDS, SOME EXTERNAL STANDARDS SUCH AS SQUIZLABS, AS WELL AS SOME OF OUR OWN

Why PHPStan?

IT INTEGRATES BETTER, ALLOWS FOR STATIC ANALYSIS, HELPS TO STRENGTHEN OUR CODE

Lets Try This Out

WHAT COULD GO WRONG WITH LIVE CODE!

This Is Only A Part Of Our Standards

We have standards that apply to every single part of our stack.

Over time these standards have evolved to cover not just how we code, but how we estimate, how we present to clients, how we manage our projects and almost every other part of our business