1x INTERNET

PHP and JavaScript are dead. Long live PHP and JavaScript

July 2023

Hristo Chonov @ 1xINTERNET GmbH

Hristo Chonov

Studied Computer Science at TU Darmstadt, Germany

Co-Maintainer for Drupal 8/9/10 Core Entity API

Author ot multiple contrib extensions for Drupal

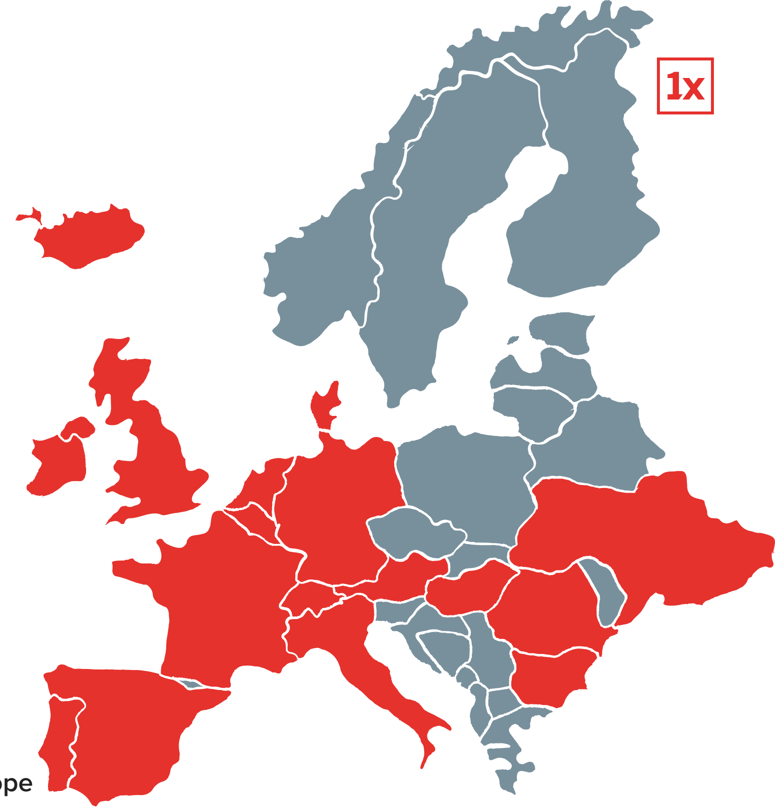
Lead Software Engineer at 1xINTERNET GmbH

We are 1xINTERNET

1xINTERNET is a full service digital agency and a leader in the development of web solutions based on Drupal and React

Our core competencies are:

1. **Digital Strategy and Consulting**
2. **User Experience and Design**
3. **Development with Drupal and React**
4. **Webtracking, SEO and Online Marketing**
5. **Hosting, Maintenance and Support Services**



International team of **75+** employees

From all corners of the world:
South America, North America, Asia and Europe

Over **23** different languages

Offices in **Frankfurt** (Headquarters), **Berlin**, **Reykjavik**,
Conil de la Frontera and Burgas (opening soon)

Founded in **2013**

PHP - History

- PHP: Hypertext Preprocessor

- Open-source scripting language, primarily designed for web development

- Created by Rasmus Lerdorf in 1994 for tracking visits to his online resume

- Released as an open-source project in 1995

PHP - Early Rapid Growth

- PHP gained popularity rapidly due to its simplicity and ease of use

- allowing developers to create dynamic websites with minimal effort

- during its growth it faced criticism and acquired a somewhat negative reputation

PHP - Early Reputation and Challenges

- Inconsistent Design: lack of a consistent and coherent design philosophy, leading to inconsistencies and quirks in the language syntax and behavior.

- Security Concerns: early versions had vulnerabilities that could be exploited if developers did not follow best security practices. This led to several high-profile security incidents, further tarnishing PHP's image.

- Poor Coding Practices: PHP's flexibility allowed developers to write code in a less structured manner, leading to spaghetti code and difficulty in maintaining large-scale applications.

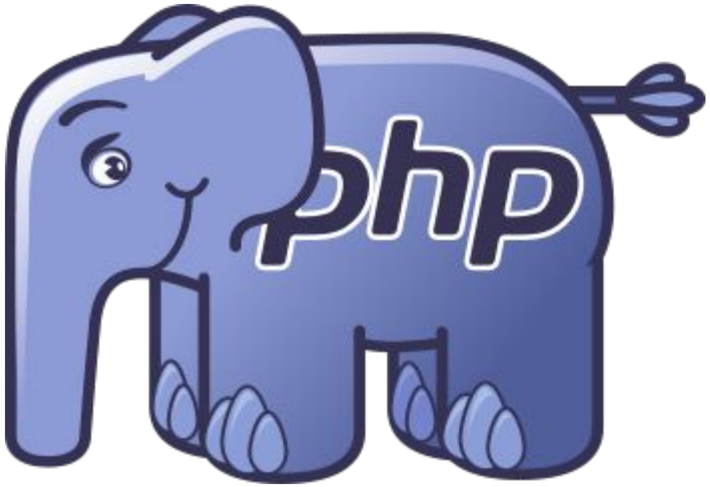
PHP - Ongoing Perception Challenges

- Legacy Codebase: Many existing PHP applications were developed using older versions and outdated practices. This legacy code can contribute to issues such as poor performance and security vulnerabilities.

- Lack of Type Safety: PHP traditionally had weak typing and lacked strong support for static typing. While newer versions introduced improvements like type hints, PHP is still perceived as being less strict compared to languages like Java or C#.

- Community Fragmentation: PHP has a large and diverse community, resulting in a wide range of coding styles, libraries, and frameworks. This fragmentation can make it challenging to navigate the PHP ecosystem and find reliable resources.

PHP - Evolution

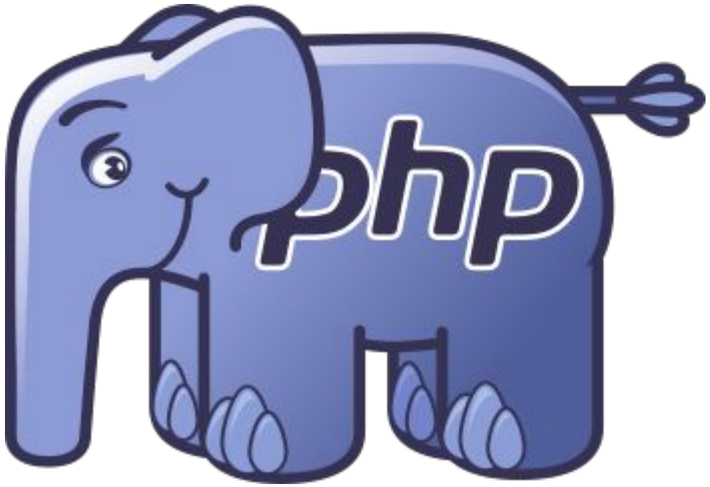


- Over the years, it has evolved into a mature and feature-rich language with a focus on performance, security, and modern development practices.

- PHP 3: Released in 1999, adding interface for multiple databases, protocols, and APIs, but the strong extensibility features attracted dozens of developers who submitted a variety of modules. Basic but limited OOP support was added.

- PHP 4: Released in 2000, a rewrite based on the Zend Engine, highly improved performance, support for many more web servers, HTTP sessions, output buffering, more secure ways of handling user input and several new language constructs such as better error handling like the try-catch block. Enhanced OOP support. Transition from passing objects by value to passing them by reference by default.

PHP 5



- PHP 5: Released in 2004, based on Zend Engine 2.0, introduced an opcode cache, that stored compiled code in memory, resulting in faster execution times, optimized object model and improved memory management.

- MySQLi - improved MySQL extension with better support for interacting with MySQL databases, including prepared statements, transactions, and improved error handling.

- XML and Web Services: built-in support for XML parsing and manipulation through the SimpleXML extension. Additionally, PHP 5 introduced the SOAP extension, enabling the creation and consumption of web services using the SOAP protocol.

- New language features, including type hinting, iterators, generators, and improved object cloning capabilities.

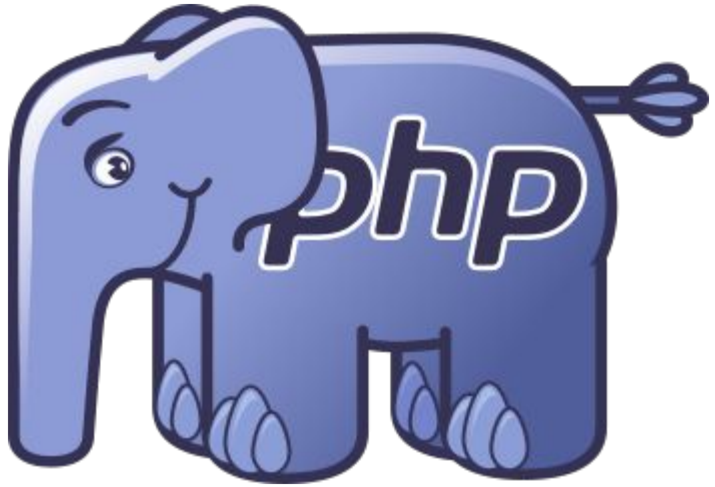
- Marked a major leap in terms of language features, performance, and object-oriented capabilities, making it a popular and widely adopted version of PHP.

PHP 7

- Released in 2015 featuring Zend Engine 3.0, providing significant performance improvements - almost twice as fast as PHP 5.6, allowing applications to handle more concurrent requests with the same hardware.

- Scalar Type Declarations: PHP 7 introduced the ability to declare the type of scalar (primitive) data types for function parameters and return values. This included support for declaring types such as int, float, string, and bool. It allowed developers to enforce stricter type checking and improve the reliability and maintainability of their code.

- Return Type Declarations: PHP 7 introduced support for declaring the return type of functions and methods. This enabled developers to specify the type of data that a function is expected to return, providing better clarity and ensuring consistency in function signatures.



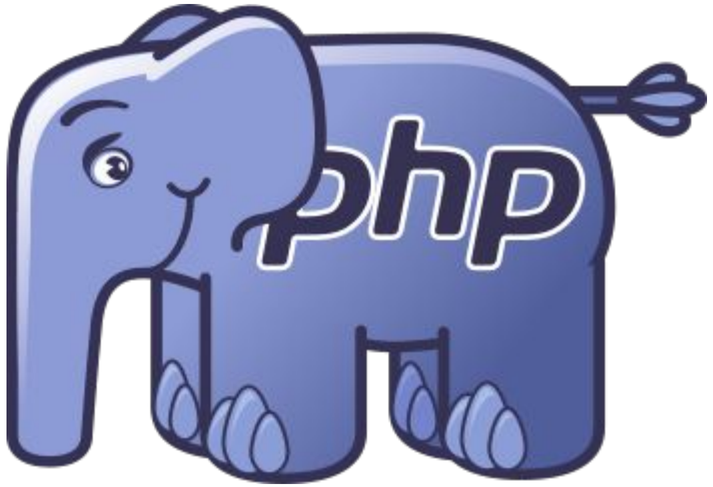
PHP 7...

- Null Coalescing Operator (??): providing a concise way to handling null values. It allowed developers to provide a default value when accessing a variable that may be null, simplifying null value checks and reducing code verbosity.

- Spaceship Operator (<=>), also known as the combined comparison operator. It allowed for simpler and more concise comparisons between two values, returning -1, 0, or 1 depending on the comparison result.

- Anonymous Classes: allowing the definition of classes on the fly without specifying a class name. This is particularly useful in scenarios where a class is needed only for a specific task or a limited scope.

- Error Handling Improvements: by introducing the Throwable interface and the Exception hierarchy. This made it easier to catch and handle both built-in and custom exceptions in a more consistent manner.



PHP 8 - another major release - 2020

Just-In-Time compilation:

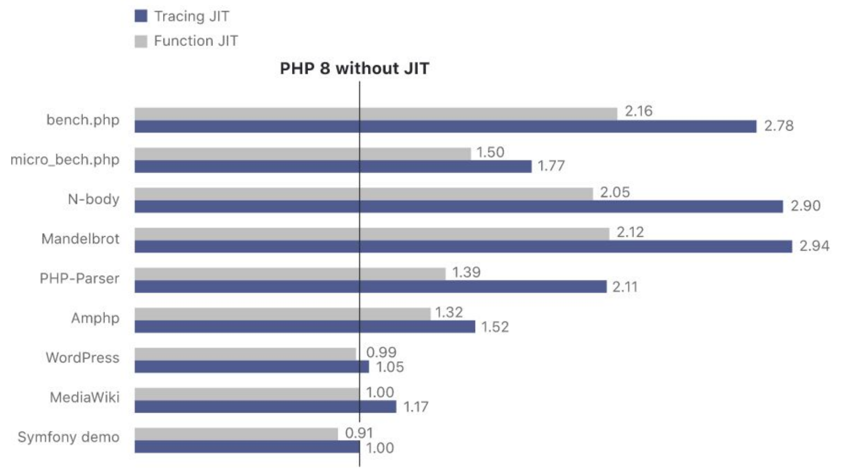
- two JIT compilation engines

- x3 better performance on synthetic benchmarks

- x1.5 - x2 improvement on some specific long-running applications

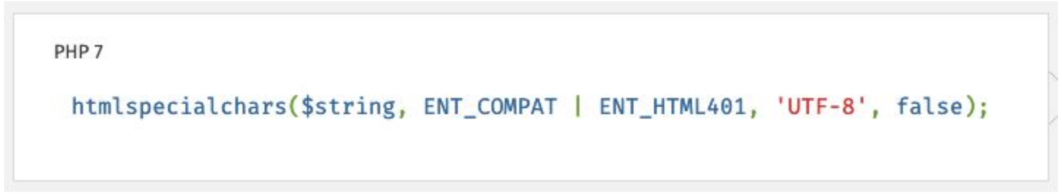
- Typical application performance is on par with PHP 7.4

Relative JIT contribution to PHP 8 performance



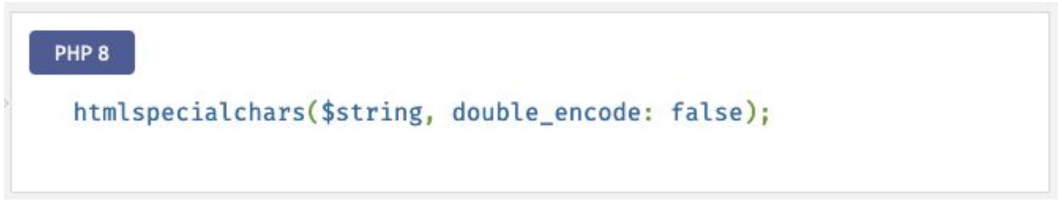
PHP 8 - Named arguments

- Specify only required parameters, skipping optional ones.
- Arguments are order-independent and self-documented.



PHP 7

```php
htmlspecialchars($string, ENT_COMPAT | ENT_HTML401, 'UTF-8', false);
```

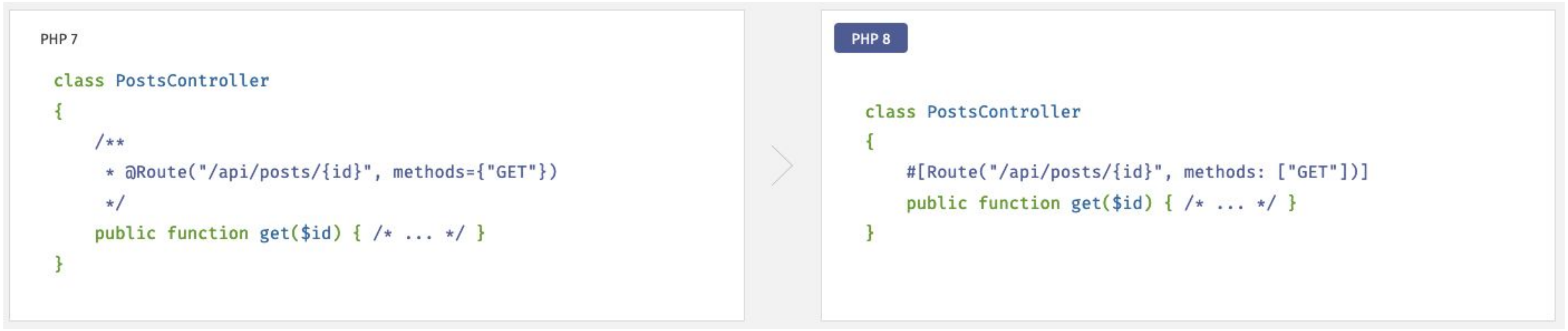


PHP 8

```php
htmlspecialchars($string, double_encode: false);
```

PHP 8 - Attributes

- structured metadata with PHP's native syntax



PHP 7

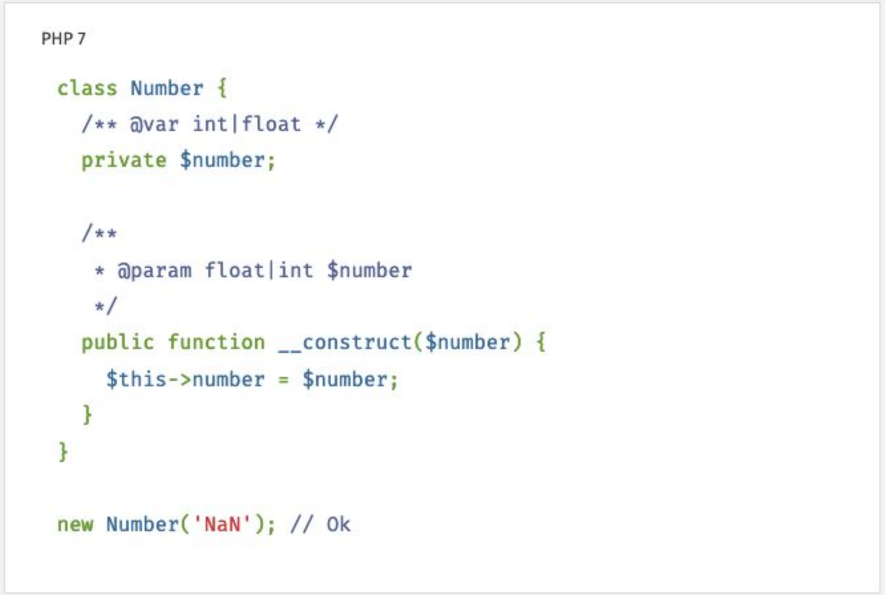
```
class PostsController
{
    /**
     * @Route("/api/posts/{id}", methods={"GET"})
     */
    public function get($id) { /* ... */ }
}
```

PHP 8

```
class PostsController
{
    #[Route("/api/posts/{id}", methods: ["GET"])]
    public function get($id) { /* ... */ }
}
```

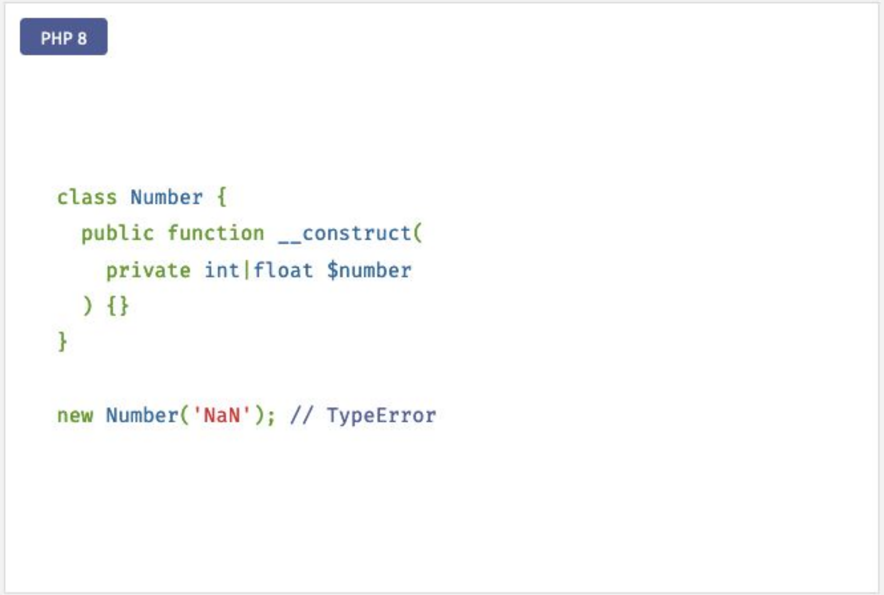
PHP 8 - Union types

- native union type declarations that are validated at runtime



PHP 7

```php
class Number {
  /** @var int|float */
  private $number;

  /**
   * @param float|int $number
   */
  public function __construct($number) {
    $this->number = $number;
  }
}

new Number('NaN'); // Ok
```



PHP 8

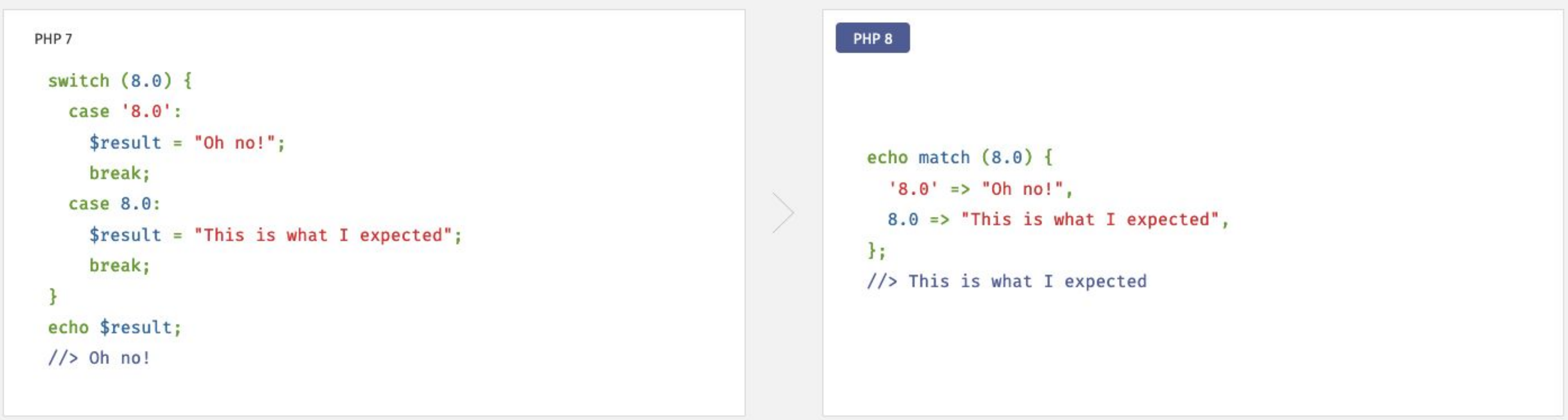
```php
class Number {
  public function __construct(
    private int|float $number
  ) {}
}

new Number('NaN'); // TypeError
```

PHP 8 - Match expression

- Match is an expression, meaning its result can be stored in a variable or returned.

- Match branches only support single-line expressions and do not need a break; statement.

- Match does strict comparisons.



PHP 7

```
switch (8.0) {
  case '8.0':
    $result = "Oh no!";
    break;
  case 8.0:
    $result = "This is what I expected";
    break;
}
echo $result;
//> Oh no!
```

PHP 8

```
echo match (8.0) {
  '8.0' => "Oh no!",
  8.0 => "This is what I expected",
};
//> This is what I expected
```

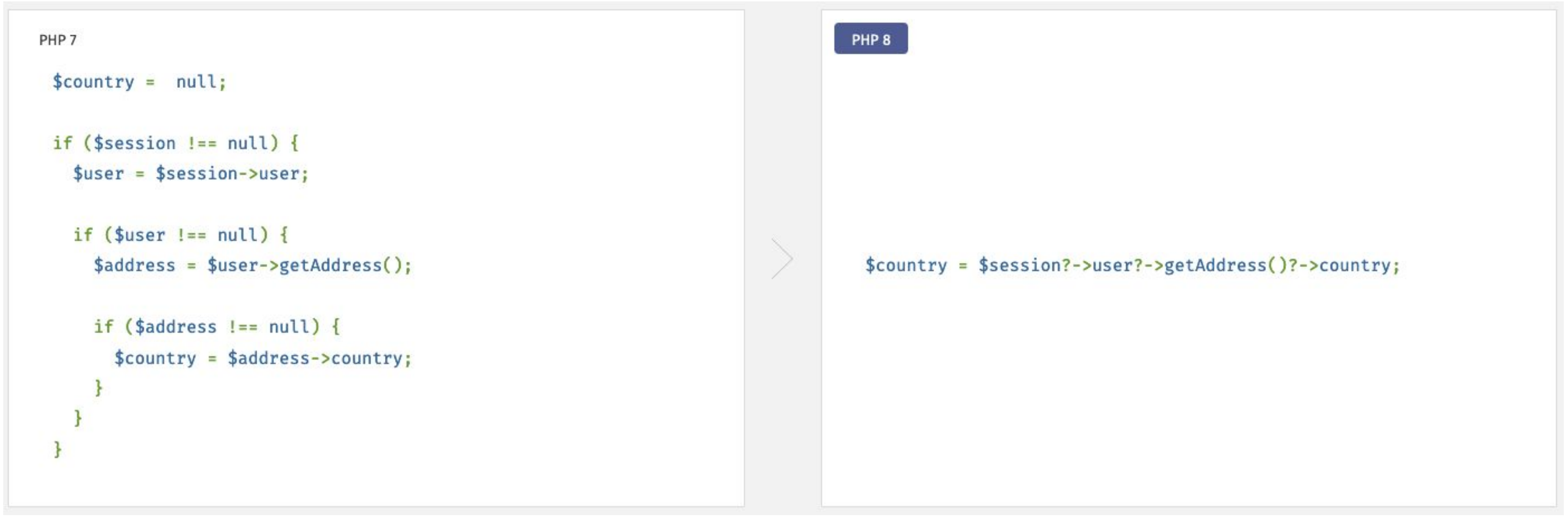
PHP 8 - Nullsafe operator

1x

- Forget about null check conditions

-> use a chain of calls with the new nullsafe operator.

- When the evaluation of one element in the chain fails, the execution of the entire chain aborts and the entire chain evaluates to null.



PHP 7

```
$country =  null;

if ($session !== null) {
  $user = $session->user;

  if ($user !== null) {
    $address = $user->getAddress();

    if ($address !== null) {
      $country = $address->country;
    }
  }
}
```

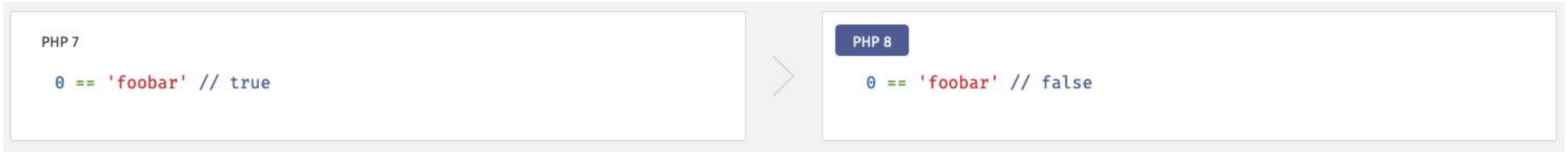
PHP 8

```
$country = $session?->user?->getAddress()?->country;
```

PHP 8 - Saner string to number comparisons

- When comparing to a numeric string, PHP 8 uses a number comparison.

- Otherwise, it converts the number to a string and uses a string comparison.



PHP 7

```
0 == 'foobar' // true
```

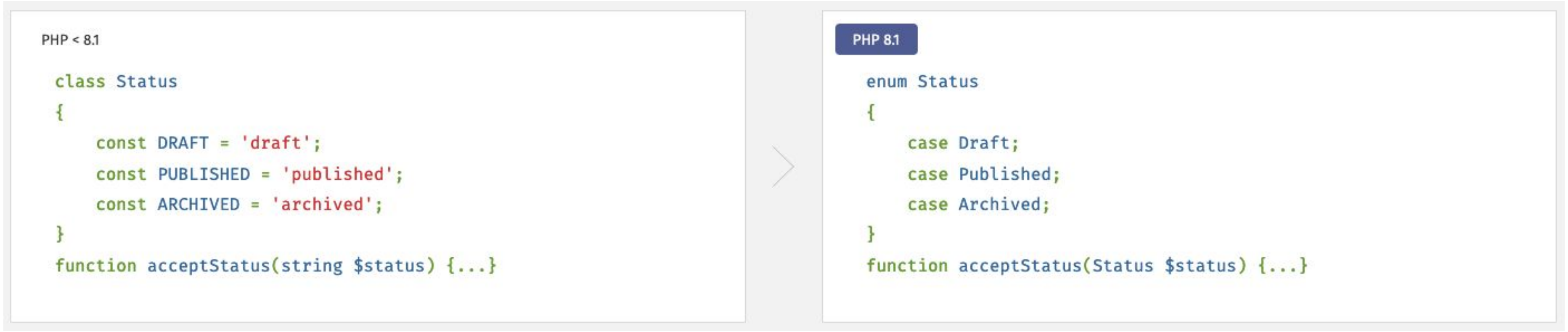
PHP 8

```
0 == 'foobar' // false
```

PHP 8.1 - 2021

Enumerations

- Use enum instead of a set of constants and get validation out of the box.



PHP < 8.1

```
class Status
{
    const DRAFT = 'draft';
    const PUBLISHED = 'published';
    const ARCHIVED = 'archived';
}
function acceptStatus(string $status) {...}
```

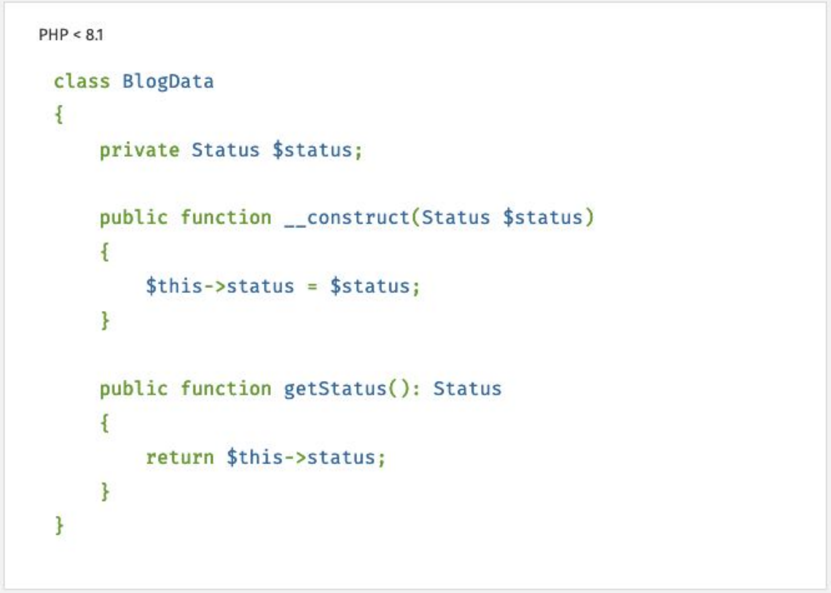
PHP 8.1

```
enum Status
{
    case Draft;
    case Published;
    case Archived;
}
function acceptStatus(Status $status) {...}
```

PHP 8.1 - Readonly Properties

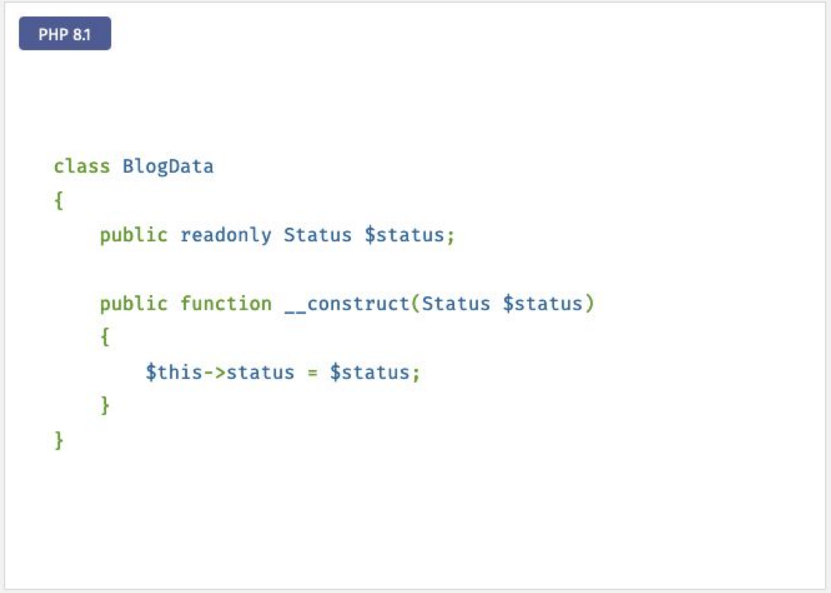
- Readonly properties cannot be changed after initialization, i.e. after a value is assigned to them.

- They are a great way to model value objects and data-transfer objects.



PHP < 8.1

```
class BlogData
{
    private Status $status;

    public function __construct(Status $status)
    {
        $this->status = $status;
    }

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

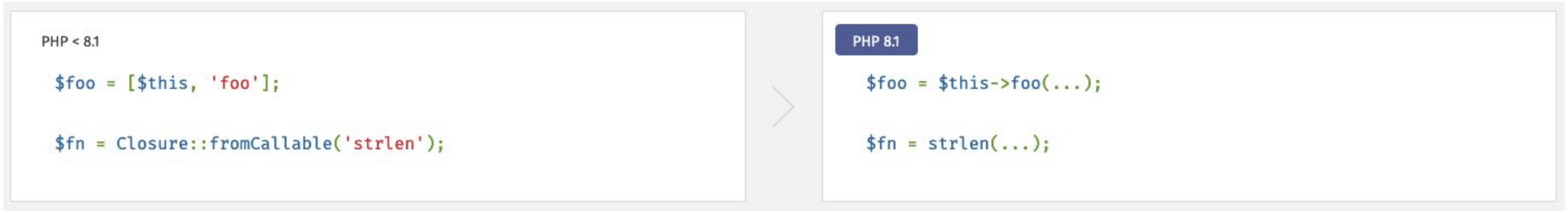


PHP 8.1

```
class BlogData
{
    public readonly Status $status;

    public function __construct(Status $status)
    {
        $this->status = $status;
    }
}
```

PHP 8.1 - First-class Callable Syntax

- It is now possible to get a reference to any function – this is called first-class callable syntax.



PHP < 8.1

```php
$foo = [$this, 'foo'];

$fn = Closure::fromCallable('strlen');
```

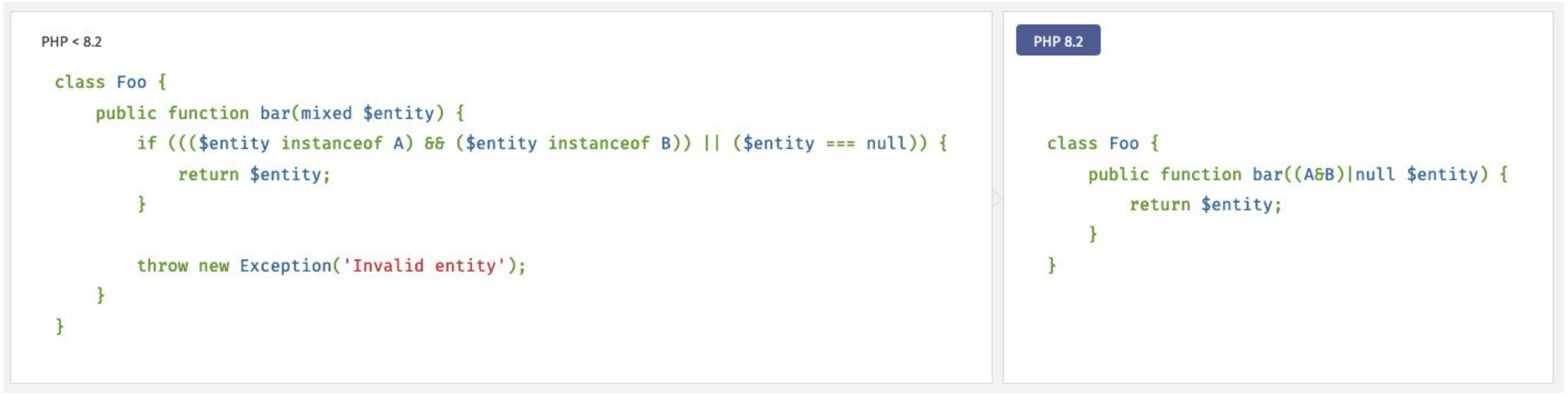
PHP 8.1

```php
$foo = $this->foo(...);

$fn = strlen(...);
```

PHP 8.2 - 2022

Disjunctive Normal Form (DNF) Types

- DNF types allow us to combine union and intersection types, following a strict rule:
  – when combining union and intersection types, intersection types must be grouped with brackets.



PHP < 8.2

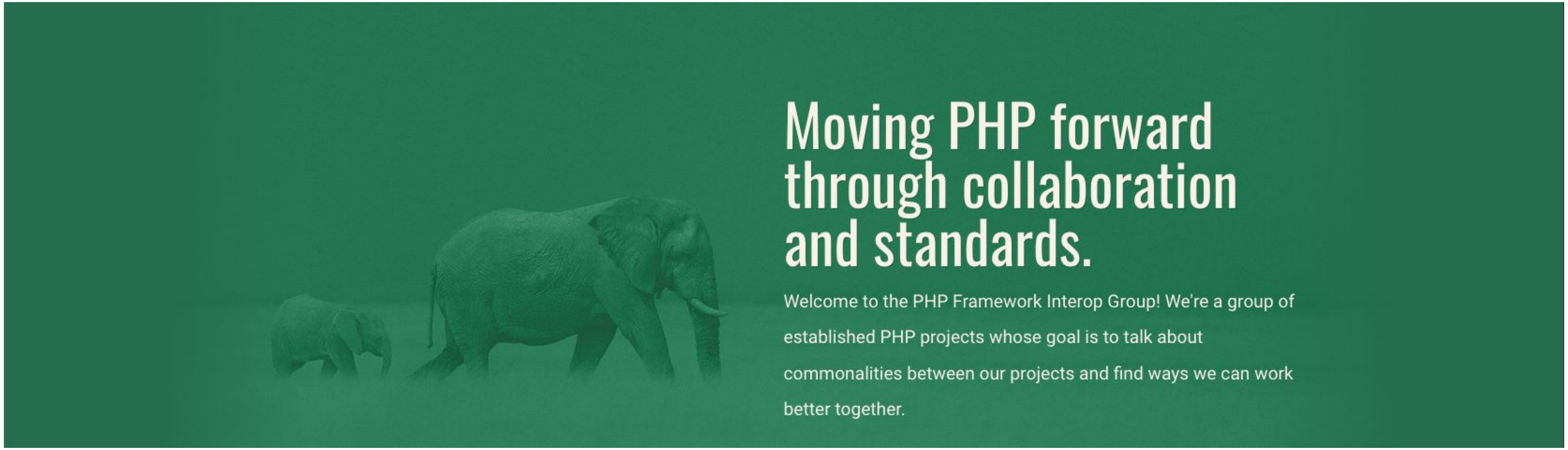
```php
class Foo {
    public function bar(mixed $entity) {
        if ((($entity instanceof A) && ($entity instanceof B)) || ($entity === null)) {
            return $entity;
        }

        throw new Exception('Invalid entity');
    }
}
```

PHP 8.2

```php
class Foo {
    public function bar((A&B)|null $entity) {
        return $entity;
    }
}
```

PHP FIG

- founded in 2009



Moving PHP forward through collaboration and standards.

Welcome to the PHP Framework Interop Group! We're a group of established PHP projects whose goal is to talk about commonalities between our projects and find ways we can work better together.

PHP FIG...

- Establish standards and best practices for PHP frameworks and packages

- Improve interoperability between different PHP frameworks and promote the reuse of software components.

- creation and adoption of PHP Standards Recommendations (PSRs).

- PSRs are a set of guidelines and specifications that define how PHP code should be written, structured, and organized to ensure compatibility and consistency across different projects.

- These standards cover various aspects of PHP development, including coding style, autoloading, logging, caching, HTTP messaging, and more.

- These PSRs, among others, help developers write more consistent, reusable, and interoperable code when working with PHP frameworks and libraries.

PHP PSRs

PSR-1: Basic Coding Standard

Defines basic coding standards that ensure a high level of interoperability between PHP projects.

PSR-2: Coding Style Guide

Specifies the coding style guide for PHP code, including formatting, indentation, and naming conventions.

PSR-3: Logger Interface

Defines a common interface for logging libraries, allowing developers to write portable code that works with different logging systems.

PSR-4: Autoloader

Describes a standardized autoloading mechanism that allows PHP classes to be automatically loaded without explicit manual inclusion.

PHP PSRs...

PSR-7: HTTP Message Interface

Provides interfaces for HTTP messages, including requests and responses, allowing different PHP frameworks to exchange HTTP messages seamlessly.

PSR-15: HTTP Server Request Handlers

Defines interfaces for request handlers and middleware, enabling the creation of reusable components for processing HTTP requests.

PSR-16: Simple Cache

Specifies a simple cache interface to standardize caching mechanisms in PHP applications.

PHPStan

- a static analysis tool for PHP code
- performs code analysis without actually executing the code
- provides insights into potential errors, type mismatches, and other issues in the codebase.
- checks for problems that can be detected at compile-time, helping developers catch bugs early in the development process.

PHPStan is needed because PHP is a dynamically typed language, which means that variable types are determined at runtime. This flexibility can lead to errors that might go unnoticed until runtime, causing unexpected behavior or crashes in the application

PHPStan...

The introduction of PHPStan has had a significant influence on PHP development and frameworks in several ways:

Increased Code Quality: PHPStan encourages developers to write cleaner and more robust code. By detecting potential bugs and type errors early on, it helps developers identify and fix issues before they cause problems in production.

Better IDE Integration: PHPStan integrates well with popular integrated development environments (IDEs) and editors, providing real-time feedback and highlighting potential issues as developers write code. This improves the developer experience and speeds up the development process.

Improved Maintainability: By enforcing strict type checking and identifying potential issues, PHPStan helps improve the maintainability of PHP codebases. It reduces the likelihood of introducing regressions or breaking changes when modifying existing code.

Influencing Frameworks and Libraries: The adoption of PHPStan has influenced the development of PHP frameworks and libraries. Many popular frameworks, such as Symfony, Laravel, and Zend Framework, have embraced PHPStan and introduced type hints and stricter typing in their codebases. This has led to more robust and predictable frameworks and improved the overall ecosystem.

Overall, PHPStan has played a crucial role in elevating the quality of PHP code, reducing bugs, and promoting best practices in PHP development. It has helped shape the PHP community's mindset towards static analysis and type checking, leading to more reliable and maintainable PHP projects.

PHP Composer

- a dependency management tool and package manager for PHP.
- It simplifies the process of managing external libraries, frameworks, and other dependencies in PHP projects.
- Composer allows developers to declare the libraries their project depends on and handles the installation, updating, and autoloading of those dependencies.

Composer is often considered simpler and easier to use compared to package managers in the other programming language's ecosystems for a few reasons:

Centralized Repository: Composer relies on the Packagist repository, which serves as a central repository for PHP packages. Packagist contains a vast collection of PHP packages that can be easily searched and added to a project. This centralized approach simplifies the process of discovering and including packages in PHP projects.

Simple Configuration: Composer uses a straightforward JSON-based configuration file called composer.json. Developers can define their project's dependencies, specify version constraints, and configure autoload settings in this file. The configuration file is easy to understand and modify, making it simpler for developers to manage dependencies.

PHP Composer...

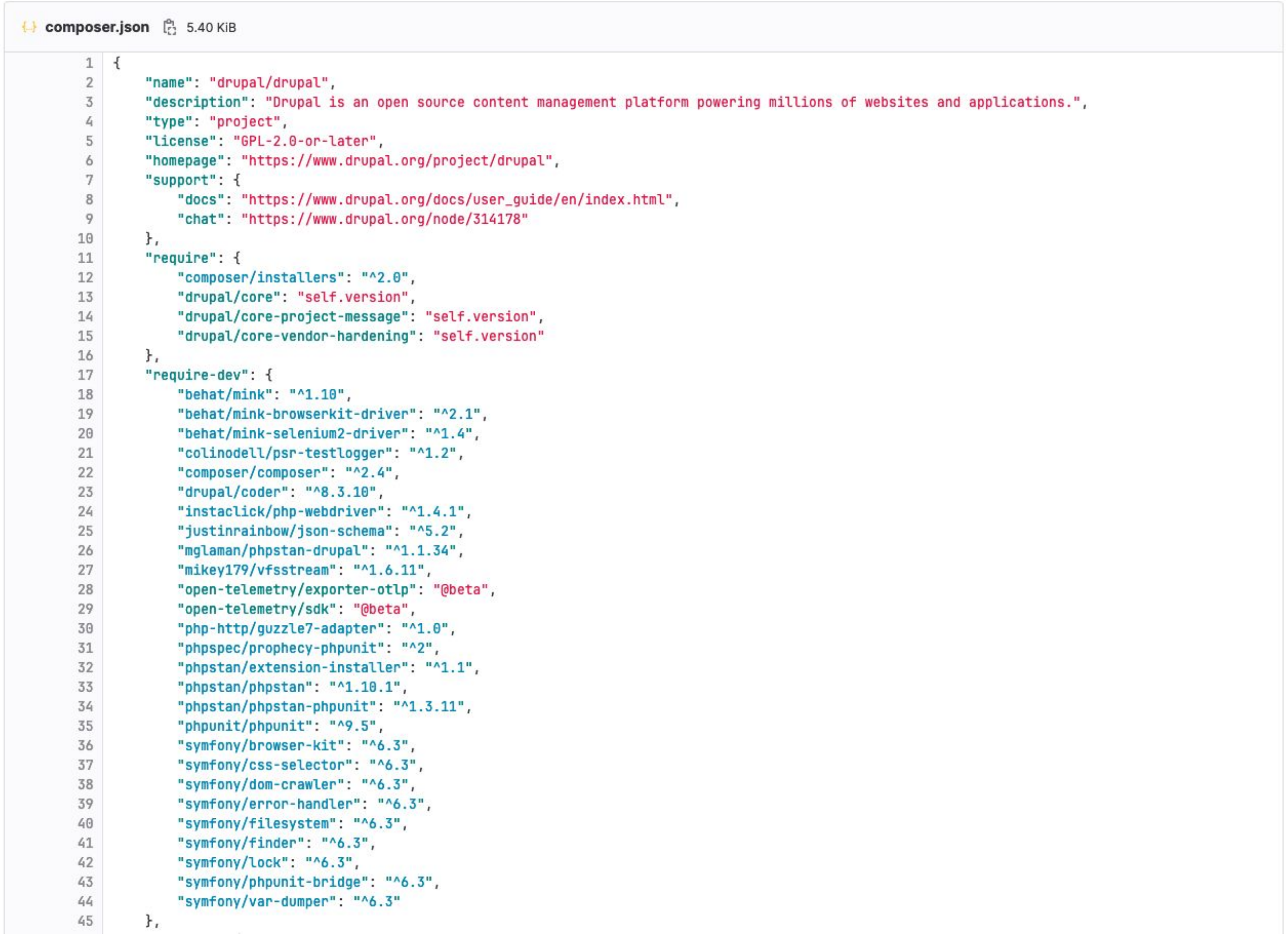
Dependency Resolution: Composer automatically resolves and manages dependencies, ensuring that the required versions of packages are installed without conflicts. It analyzes the dependencies specified in the composer.json file and determines the compatible versions to install. Composer handles complex dependency graphs effectively, simplifying the process of managing dependencies in PHP projects.

Dependency Autoloading: Composer provides a built-in autoloader that simplifies the inclusion of dependencies in PHP code. It generates an optimized autoloader based on the project's dependencies, allowing developers to access classes and functions from included packages without manual require/include statements.

Command-Line Interface (CLI): Composer offers a command-line interface with a range of commands for managing dependencies, including installation, updating, and removal of packages. The CLI provides a straightforward and consistent way to interact with Composer, making it easy to integrate into development workflows and build scripts.

PHP Composer...

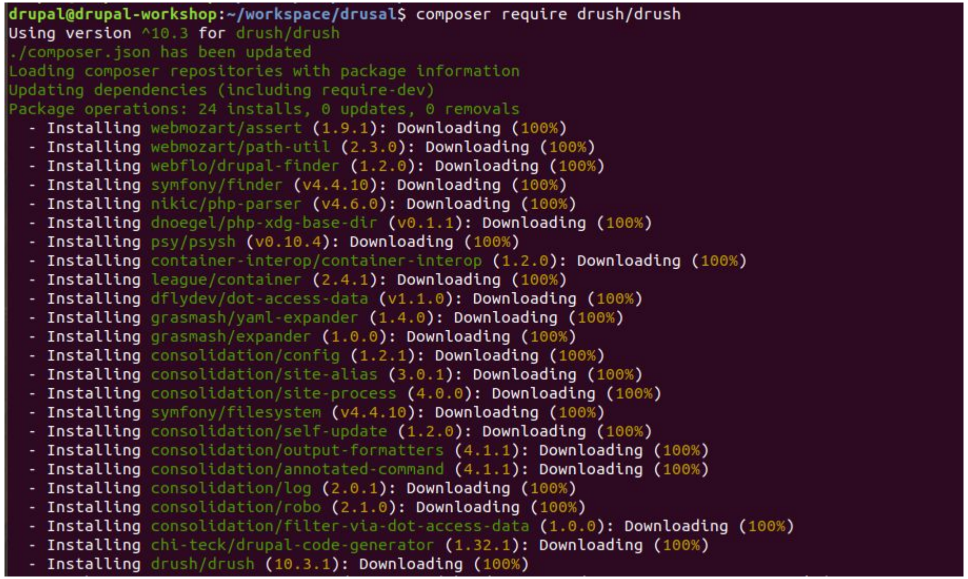
1x


composer.json 5.40 KiB

```
{
    "name": "drupal/drupal",
    "description": "Drupal is an open source content management platform powering millions of websites and applications.",
    "type": "project",
    "license": "GPL-2.0-or-later",
    "homepage": "https://www.drupal.org/project/drupal",
    "support": {
        "docs": "https://www.drupal.org/docs/user_guide/en/index.html",
        "chat": "https://www.drupal.org/node/314178"
    },
    "require": {
        "composer/installers": "^2.0",
        "drupal/core": "self.version",
        "drupal/core-project-message": "self.version",
        "drupal/core-vendor-hardening": "self.version"
    },
    "require-dev": {
        "behat/mink": "^1.10",
        "behat/mink-browserkit-driver": "^2.1",
        "behat/mink-selenium2-driver": "^1.4",
        "colinodell/psr-testlogger": "^1.2",
        "composer/composer": "^2.4",
        "drupal/coder": "^8.3.10",
        "instaclick/php-webdriver": "^1.4.1",
        "justinrainbow/json-schema": "^5.2",
        "mglaman/phpstan-drupal": "^1.1.34",
        "mikey179/vfsstream": "^1.6.11",
        "open-telemetry/exporter-otlp": "@beta",
        "open-telemetry/sdk": "@beta",
        "php-http/guzzle7-adapter": "^1.0",
        "phpspec/prophecy-phpunit": "^2",
        "phpstan/extension-installer": "^1.1",
        "phpstan/phpstan": "^1.10.1",
        "phpstan/phpstan-phpunit": "^1.3.11",
        "phpunit/phpunit": "^9.5",
        "symfony/browser-kit": "^6.3",
        "symfony/css-selector": "^6.3",
        "symfony/dom-crawler": "^6.3",
        "symfony/error-handler": "^6.3",
        "symfony/filesystem": "^6.3",
        "symfony/finder": "^6.3",
        "symfony/lock": "^6.3",
        "symfony/phpunit-bridge": "^6.3",
        "symfony/var-dumper": "^6.3"
    },
```

PHP Composer...

For example while the Java ecosystem has its own package managers like Maven and Gradle, they often have more complex configurations and conventions. They require additional XML or Groovy configuration files, and the dependency management can be more verbose and intricate, especially for larger projects. In contrast, Composer's simplicity and focus on PHP-specific requirements make it more approachable and user-friendly for PHP developers.



```
drupal@drupal-workshop:~/workspace/drusal$ composer require drush/drush
Using version ^10.3 for drush/drush
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 24 installs, 0 updates, 0 removals
  - Installing webmozart/assert (1.9.1): Downloading (100%)
  - Installing webmozart/path-util (2.3.0): Downloading (100%)
  - Installing webflo/drupal-finder (1.2.0): Downloading (100%)
  - Installing symfony/finder (v4.4.10): Downloading (100%)
  - Installing nikic/php-parser (v4.6.0): Downloading (100%)
  - Installing dnoegel/php-xdg-base-dir (v0.1.1): Downloading (100%)
  - Installing psy/psysh (v0.10.4): Downloading (100%)
  - Installing container-interop/container-interop (1.2.0): Downloading (100%)
  - Installing league/container (2.4.1): Downloading (100%)
  - Installing dflydev/dot-access-data (v1.1.0): Downloading (100%)
  - Installing grasmash/yaml-expander (1.4.0): Downloading (100%)
  - Installing grasmash/expander (1.0.0): Downloading (100%)
  - Installing consolidation/config (1.2.1): Downloading (100%)
  - Installing consolidation/site-alias (3.0.1): Downloading (100%)
  - Installing consolidation/site-process (4.0.0): Downloading (100%)
  - Installing symfony/filesystem (v4.4.10): Downloading (100%)
  - Installing consolidation/self-update (1.2.0): Downloading (100%)
  - Installing consolidation/output-formatters (4.1.1): Downloading (100%)
  - Installing consolidation/annotated-command (4.1.1): Downloading (100%)
  - Installing consolidation/log (2.0.1): Downloading (100%)
  - Installing consolidation/robo (2.1.0): Downloading (100%)
  - Installing consolidation/filter-via-dot-access-data (1.0.0): Downloading (100%)
  - Installing chi-teck/drupal-code-generator (1.32.1): Downloading (100%)
  - Installing drush/drush (10.3.1): Downloading (100%)
```

PHP Frameworks

Symfony, Drupal, and Laravel are among the most famous and widely used PHP frameworks. While they serve different purposes and have distinct characteristics, there are connections and commonalities among them.

1x

PHP Frameworks - Symfony

Symfony is a powerful and widely-used PHP framework known for its flexibility, scalability, and extensive set of reusable components. These components, also known as Symfony Components, are standalone libraries that provide specific functionalities and can be used independently of the Symfony framework itself. They follow best practices, adhere to PHP Standards Recommendations (PSRs), and promote code reusability.

Flex: Compose your Application
- amazing tool that makes adding new features as simple as running one command
- it's also the reason why Symfony is ideal for a small micro-service or a huge application.

https://symfony.com/doc/current/quick_tour/flex_recipes.html

Symfony

1x

Symfony - Components

HttpFoundation:
HttpFoundation provides an object-oriented layer for handling HTTP requests and responses. It encapsulates request information (headers, parameters, cookies, etc.) and provides an easy-to-use API for building and manipulating HTTP responses.

Routing:
The Routing component offers a flexible system for defining and matching URLs to controllers and actions. It allows developers to define routes and their corresponding handlers, making it easier to implement clean and SEO-friendly URLs.

HttpKernel:
HttpKernel is responsible for managing the lifecycle of an HTTP request in a Symfony application. It handles the processing of requests, dispatching them to appropriate controllers, and managing events and middlewares during the request-response cycle.

DependencyInjection:
The DependencyInjection component provides a powerful inversion of control (IoC) container that allows developers to manage and resolve dependencies in a flexible manner. It simplifies the management of services and their configurations, promoting loose coupling and modularity.

Symfony

Symfony - Components

Form:
The Form component simplifies the creation, handling, and validation of HTML forms. It provides a convenient way to define form fields, handle user input, and perform validation and data transformation.

Validator:
Validator is a component that provides a comprehensive set of tools for data validation. It offers a wide range of constraints and validation rules that can be applied to ensure the correctness and integrity of user-submitted data.

Security:
The Security component provides tools for handling authentication, authorization, and secure user management. It offers features such as user authentication, access control, firewall configuration, and encryption.

Translation:
The Translation component assists in the translation and localization of applications. It provides a convenient way to manage and translate messages in multiple languages.

PHP Frameworks - Drupal

Drupal is an open-source content management framework (CMF) built on PHP and Symfony components. It offers a powerful platform for building websites, online communities, and web applications. Drupal follows a modular architecture and provides a flexible and customizable framework for managing content, user permissions, and site functionality. Drupal is known for its robustness, scalability, and extensive customization capabilities.

Drupal leverages Symfony components, such as the HTTP Foundation and Dependency Injection components, to handle crucial aspects of the framework. The integration of Symfony components allows Drupal to benefit from the robustness, security, and interoperability of the Symfony ecosystem.

Install Drupal in 5 Minutes:
https://www.youtube.com/watch?v=pYCZixiXX2U

- Install DDEV: https://ddev.readthedocs.io/en/latest/users/install/ddev-installation/
- Install Drupal: https://ddev.readthedocs.io/en/latest/users/quickstart#drupal

Drupal

Modular Architecture:
Drupal follows a modular architecture, allowing developers to extend its functionality through modules. Modules provide additional features and can be installed, enabled, and configured to meet specific requirements. Drupal has a vast ecosystem of contributed modules, providing a wide range of functionalities that can be integrated seamlessly into websites.

Content Management Features:
Drupal offers comprehensive content management features, making it suitable for websites with extensive content requirements. It provides a flexible content modeling system, allowing administrators to define custom content types, fields, and taxonomies. Drupal's content management capabilities include versioning, workflow management, content revisioning, and advanced permission controls.

Scalability and Performance:
Drupal is designed to handle high-traffic and large-scale websites. It has built-in caching mechanisms, database optimization, and performance tuning options to ensure optimal performance. Drupal's modular approach allows developers to scale websites by adding or optimizing specific modules based on their needs.

Drupal

Flexibility and Customization:
Drupal provides a highly flexible and customizable platform. It allows developers to create custom themes, layouts, and templates to achieve unique visual designs. The flexible theming system and extensive template options provide full control over the appearance of the website. Additionally, Drupal's powerful API allows for seamless integration with external systems and services.

Security and Community:
Drupal has a strong focus on security and follows best practices to mitigate security risks. The Drupal Security Team actively monitors and addresses security vulnerabilities promptly. Furthermore, Drupal has a large and active community of developers, designers, and contributors who continuously contribute to its development, security, and documentation. The community provides support, shares knowledge, and contributes to the improvement and evolution of the Drupal platform.

Multilingual Support:
Drupal offers robust multilingual support out of the box. It enables the creation of multilingual websites, allowing content to be translated into multiple languages. Drupal provides language handling features, translation workflows, and localization capabilities to cater to diverse language requirements.

Integration Capabilities:
Drupal can integrate with various third-party systems, services, and APIs. It offers APIs for content sharing, authentication, and data exchange, enabling seamless integration with external applications and services.

1x

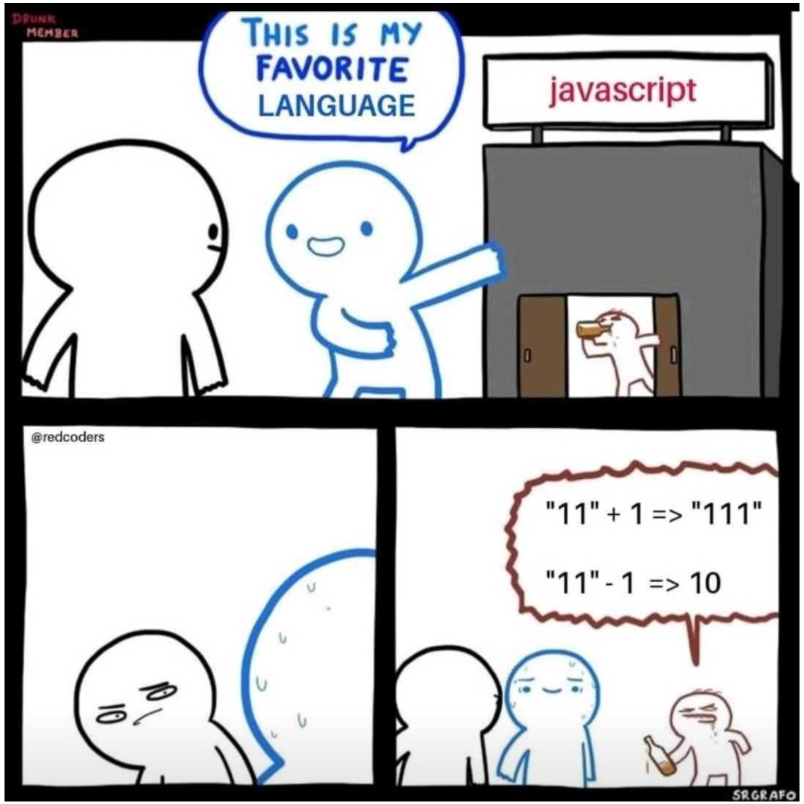
JavaScript - History

Creation and Early Days:

JavaScript was created by Brendan Eich at Netscape Communications in 1995. Initially called "Mocha," it was later renamed to "LiveScript" and eventually settled on "JavaScript" to leverage the popularity of Java at that time. It was primarily designed as a scripting language for client-side web development, allowing developers to add interactivity and dynamic features to websites.

Standardization and Browser Wars:

As the web grew in popularity, different browser vendors implemented their own versions of JavaScript, leading to inconsistencies and compatibility issues. To address this, the European Computer Manufacturers Association (ECMA) created a standardization process for JavaScript, resulting in the first ECMAScript specification (ES1) in 1997.



JavaScript - History

Rise of Ajax and Web 2.0:
With the emergence of Ajax (Asynchronous JavaScript and XML) in the early 2000s, JavaScript gained significant attention. Ajax enabled seamless data exchange between the browser and server, allowing web applications to provide dynamic and interactive user experiences. JavaScript became an integral part of the Web 2.0 era, powering modern web applications.

JavaScript in the Browser and Beyond:
JavaScript's popularity continued to soar as browser vendors improved performance and introduced new features. JavaScript was no longer limited to the browser; it started being used on the server-side as well. Projects like Node.js, introduced in 2009, enabled JavaScript to be used as a full-stack development language, further expanding its reach.

Evolving Language and Community:
Over the years, JavaScript has undergone significant advancements. ECMAScript standards continued to evolve with regular updates. New features like arrow functions, classes, modules, and async/await were introduced, improving the language's expressiveness and capabilities. The JavaScript community expanded, with numerous libraries, frameworks, and tools emerging, catering to various development needs.

JavaScript - becoming mature

Language Enhancements:
The introduction of new ECMAScript standards and language features has made JavaScript more powerful, expressive, and easier to work with. It has become more structured and supports modern programming paradigms.

Robust Ecosystem:
The JavaScript ecosystem has grown tremendously, with a vast collection of libraries, frameworks, and tools. This allows developers to build complex, scalable, and maintainable applications more efficiently. Frameworks like Angular, React, and Vue.js have gained popularity for enterprise development.

Performance Improvements:
JavaScript engines in modern browsers have significantly improved performance, making JavaScript execution faster than ever. Just-in-Time (JIT) compilation and optimization techniques have enhanced runtime performance.

Tooling and Development Practices:
Developers now have access to a wide range of powerful development tools, integrated development environments (IDEs), code editors, testing frameworks, and build tools. This has improved productivity, code quality, and developer experience.

Community Support:
The JavaScript community is vibrant, active, and supportive. Online communities, conferences, and open-source contributions have fostered knowledge sharing and collaboration, leading to advancements and best practices in JavaScript development.

JavaScript - TypeScript

TypeScript is a programming language and open-source superset of JavaScript developed by Microsoft and released in 2012. It builds upon JavaScript by adding optional static typing, advanced tooling support, and modern language features.

Static Typing:
TypeScript introduces static typing to JavaScript, allowing developers to explicitly declare variable types, function signatures, and object structures. This enables early detection of errors during development, improves code quality, and enhances the reliability of JavaScript applications. Static typing also provides better code documentation and enables IDEs to offer enhanced code suggestions, autocompletion, and refactoring tools.

Enhanced Tooling:
TypeScript comes with a rich set of development tools, including a robust compiler and an integrated development environment (IDE) support. The TypeScript compiler translates TypeScript code into plain JavaScript, making it compatible with all JavaScript environments. IDEs like Visual Studio Code provide excellent TypeScript support with features like IntelliSense, type checking, and code navigation, which greatly enhance developer productivity.

Modern Language Features:
TypeScript supports modern language features like classes, modules, interfaces, and lambda functions. It brings object-oriented programming (OOP) concepts to JavaScript, making it easier to write and maintain complex applications. TypeScript also supports ECMAScript standards, so developers can leverage the latest JavaScript features and syntax while enjoying the benefits of static typing.

JavaScript - TypeScript

Improved Collaboration:
TypeScript facilitates collaboration in larger development teams. With explicit types and interfaces, it provides better communication and understanding of code among team members. TypeScript's static typing helps catch common errors and reduces the likelihood of introducing bugs during development. This leads to more maintainable and scalable codebases, making it easier for developers to work together on JavaScript projects.

Gradual Adoption:
TypeScript allows for gradual adoption in existing JavaScript projects. It supports mixed JavaScript and TypeScript codebases, enabling developers to introduce TypeScript incrementally. This flexibility allows organizations to leverage TypeScript's benefits without rewriting their entire codebase.

Strong Ecosystem and Community:
TypeScript has gained significant popularity and has a strong ecosystem of libraries and frameworks that provide TypeScript support. Popular frameworks like Angular and React have embraced TypeScript as their primary language, providing excellent integration and tooling for TypeScript developers. The vibrant TypeScript community actively contributes to its development, creates libraries, and shares best practices.

TypeScript has helped JavaScript evolve by addressing some of its limitations and making it more scalable, maintainable, and reliable. It has become a popular choice for large-scale applications, enterprise projects, and collaborations, bridging the gap between JavaScript's dynamic nature and the benefits of static typing and tooling support.

JavaScript - Frameworks

React, Vue, Angular, and Next.js are popular JavaScript frameworks and libraries that have significantly contributed to the evolution of JavaScript and web development. Here's a brief description of each and how they have helped JavaScript evolve:

React is a JavaScript library developed by Facebook. It enables the creation of reusable UI components and provides a declarative approach to building user interfaces. React introduced the concept of a virtual DOM (Document Object Model), allowing efficient updates to the UI by minimizing direct DOM manipulations. React's component-based architecture and one-way data flow made it easier to build complex UIs, resulting in improved code reusability and maintainability. React's influence has led to the development of similar component-based frameworks and libraries.

Next.js is a framework built on top of React that enables server-side rendering (SSR) and static site generation (SSG) for React applications. It simplifies the process of building performant and SEO-friendly web applications. Next.js offers features like automatic code splitting, server-side rendering, and API routes, making it ideal for building hybrid server-rendered and client-side rendered applications. Next.js has played a significant role in advancing React-based development practices and making server-side rendering more accessible.

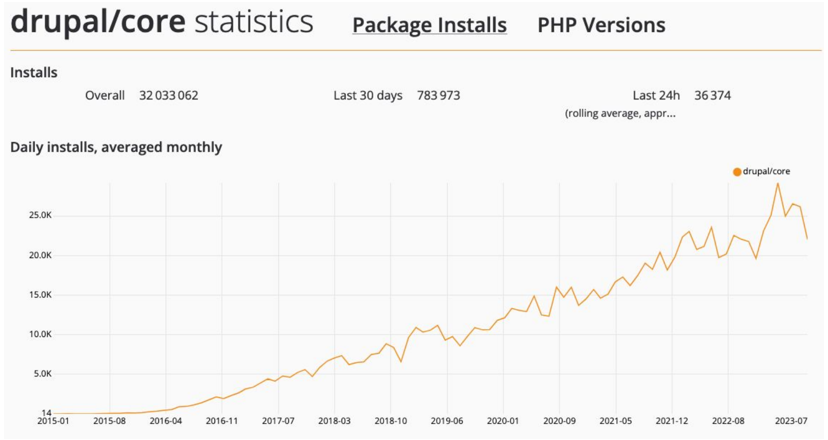
JavaScript - Frameworks

Angular is a full-featured JavaScript framework developed by Google. It offers a comprehensive set of tools and features for building large-scale applications. Angular follows a component-based architecture and uses a two-way data binding approach. It provides powerful features like dependency injection, TypeScript support, and a complete development ecosystem. Angular's focus on strong architectural patterns and scalability has made it suitable for enterprise-level applications.

Vue is a progressive JavaScript framework designed for building user interfaces. It combines ease of use with powerful features. Vue provides a flexible architecture, allowing developers to incrementally adopt its features in existing projects. Vue's reactivity system enables efficient data binding, making it easy to create dynamic and responsive UIs. Vue's simplicity and approachability have attracted a large and passionate community, leading to its widespread adoption.

1x

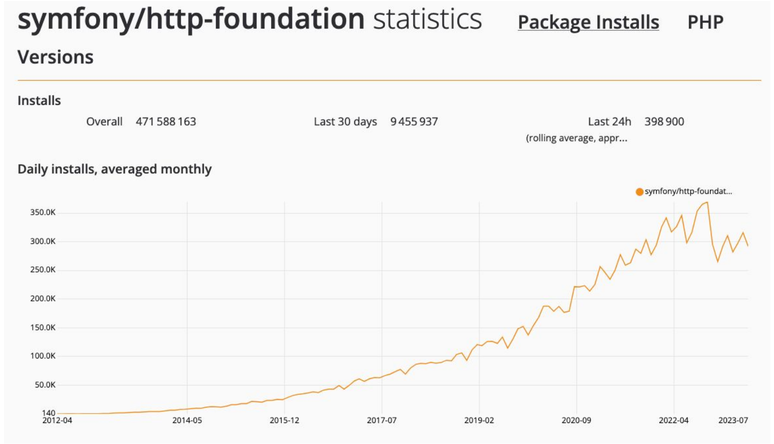
When did PHP actually die?


drupal/core statistics Package Installs PHP Versions

Installs

Overall 32 033 062 Last 30 days 783 973 Last 24h 36 374 (rolling average, appr...

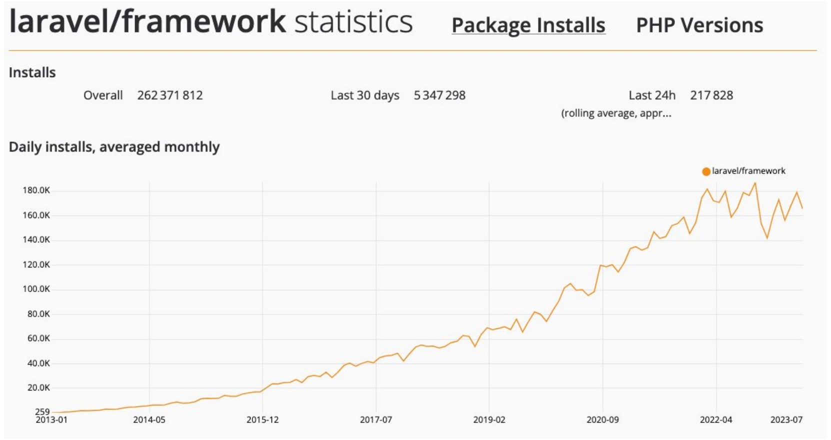
Daily installs, averaged monthly


symfony/http-foundation statistics Package Installs PHP Versions

Installs

Overall 471 588 163 Last 30 days 9 455 937 Last 24h 398 900 (rolling average, appr...

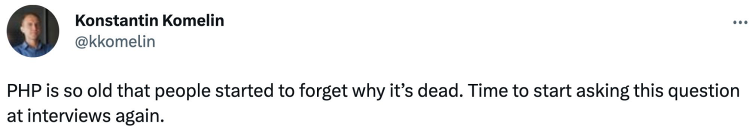
Daily installs, averaged monthly


laravel/framework statistics Package Installs PHP Versions

Installs

Overall 262 371 812 Last 30 days 5 347 298 Last 24h 217 828 (rolling average, appr...

Daily installs, averaged monthly


Konstantin Komelin
@kkomelin

PHP is so old that people started to forget why it's dead. Time to start asking this question at interviews again.

Demo: Drupal + Next.js

1. mkdir umami-drupal && cd umami-drupal
2. ddev config --project-type=drupal10 --docroot=web --create-docroot
3. ddev start
4. ddev composer create drupal/recommended-project
5. Edit composer.json and add this to the “extra” section:

```
  "patches": {
    "drupal/subrequests": {
      "Get same results on different request": "https://www.drupal.org/files/issues/2019-07-18/change_request_type-63049395-09.patch"
  },
    "drupal/decoupled_router": {
      "Unable to resolve path on node in other language than default":
"https://www.drupal.org/files/issues/2023-04-27/decouple_router-3111456-resolve-language-issue-64--get-translation.patch"
    }
  }
```

8. composer require drupal/next drupal/jsonapi_menu_items drupal/jsonapi_views 'drupal/jsonapi_resources:^1.0@beta' -W
9. ddev launch
10. Install Drupal profile “Umami Demo”
11. Enable the following modules:
  - Next
  - Next JSON:API
  - JSON:API Menu Items
  - JSON:API Views

https://ddev.readthedocs.io/en/latest/users/install/ddev-installation/
https://ddev.readthedocs.io/en/latest/users/quickstart/
https://next-drupal.org/learn/quick-start/install-drupal
https://next-drupal.org/guides/umami-demo

Demo: Drupal + Next.js

12. git clone https://github.com/chapter-three/next-example-umami umami-next
13. cd umami-next
14. nvm use node 16
15. yarn install
16. cp .env.example .env.local
17. Add information about the backend to the local env of the Next.js
    - echo -e “\nNEXT_PUBLIC_DRUPAL_BASE_URL=https://umami-drupal.ddev.site\nNEXT_IMAGE_DOMAIN=umami-drupal.ddev.site\nNODE_TLS_REJECT_UNAUTHORIZED=0” >> .env.local
18. yarn dev

https://ddev.readthedocs.io/en/latest/users/install/ddev-installation/
https://ddev.readthedocs.io/en/latest/users/quickstart/
https://next-drupal.org/learn/quick-start/install-drupal
https://next-drupal.org/guides/umami-demo

We are hiring!
jobs@1xinternet.de

1xINTERNET GmbH
Seelenberger Str. 2
60489 Frankfurt (D)

+49 69 976 711 50
office@1xinternet.de
www.1xinternet.de