Architecting Web Applications using PHP

Session 1

Introduction to PHP

Session Overview

In this session, you will learn to:

- Describe the evolution of PHP
- List the important milestones in the development of PHP
- List the enhancements in PHP 8.0
- Outline JIT Compilation
- List the features and uses of PHP 8.0

History of PHP

PHP is a successor of PHP/FI Product, the history of which, dates back to the year 1994.

PHP Version	Release Date
1.0	October 1995
2.0	April 1996
3.0	June 1997
4.0	June 1999
5.0	July 2004
6.0	
7.0	2019
8.0	2020

Table 1.1: Different Versions of PHP

Introduction to Hypertext Preprocessor (PHP)

What is PHP?

Is a widely-used general-purpose, open source, and server-side scripting language, especially suited for Web development.

Can be written and saved as a PHP script or embedded into HTML.

Is used for developing client-side Graphical User Interface (GUI) applications that are platform-independent, and for command-line scripting as well.

Features of PHP [1-2]

Some well-known and important features of PHP:

- Relatively easy for a fresher and also has many advanced features for a professional programmer.
- Runs efficiently on the server-side.
- Works on many operating systems such as Linux, Windows, and Mac OS X.
- Is free and can be downloaded from official PHP resource: www.php.net
- Supports many databases such as Oracle, MySQL, MS SQL Server, Sybase, and so on.

1.3 More Features of PHP [2-2]

PHP can dynamically generate content of type HTML, PDF, Text, XML, CSV, and many others.

Writing programs in PHP is easy and fast, which effectively means that it takes less time to build an application.

Many popular PHP frameworks such as Zend, Laravel, and CodeIgniter are available for PHP.

There are several Web hosting options available at a fair price for PHP.

Code deployment is straightforward with PHP.

Uses of PHP

Areas in which PHP scripts are mainly used:

Server-side Scripting: It requires a PHP parser, a Web server, and a Web browser. Software packages such as XAMPP are required for server-side scripting.

Command-line Scripting: It allows PHP scripts to run without any server or browser. A PHP parser is required for command-line scripting.

Writing Desktop: If a user has ample knowledge of PHP and is using advanced PHP features in their applications, they can use PHP-GTK to write their programs.

Basic PHP Syntax

Code Snippet 1:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Example</title>
  </head>
  <body>
    <?php
      echo "Hello Students!";
    ?>
  </body>
</html>
```

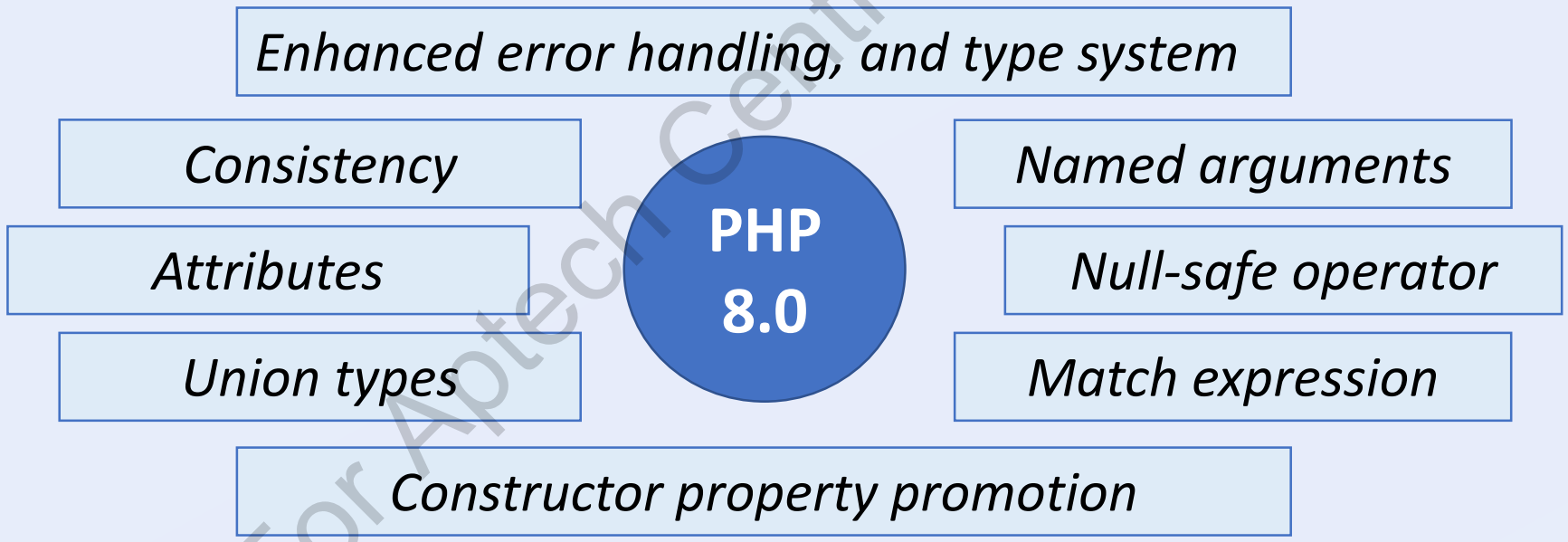
Output: Hello Students!

Difference between PHP and JavaScript

PHP	JavaScript
PHP is a popular scripting language which is used to perform server-side functions.	JavaScript is a high level programming language which is handy for client-side scripting
In PHP, the code is executed on server, not within browser.	In JavaScript, the code is executed within browser (client) itself.
PHP is a back-end language.	JavaScript is used for front-end development.
PHP can only be embedded with HTML.	JavaScript can be combined with HTML, AJAX, XML.
PHP is a multi-threaded language. It means that it blocks I/O to carry out multiple tasks concurrently.	JavaScript is a single-threaded language which is event driven. It never blocks everything and runs concurrently.
PHP helps to build high-level interactive web pages.	JavaScript helps to build user-friendly creative Web pages.

Enhancements in PHP 8.0

All new PHP 8.0 comes with various new features and optimizations are mentioned as follows:



Named Arguments

Following representation shows how to pass a Named argument to a function in PHP 7.0 and PHP 8.0:

PHP 7.0

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

PHP 8.0

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

Attributes

Following representation shows the syntax for attributes in PHP 7.0 and PHP 8.0 respectively:

PHP 7.0

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

PHP 8.0

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

Constructor Property Promotion

Following representation shows the example for constructor property in PHP 7.0 and PHP 8.0 respectively:

PHP 7.0

```
class Point {
  public float $x;
  public float $y;
  public float $z;

  public function __construct(
   float $x = 0.0,
   float $y = 0.0,
   float $z = 0.0
  ) {
   $this->x = $x;
   $this->y = $y;
   $this->z = $z;
  }
}
```

PHP 8.0

```
class Point {
  public function __construct(
    public float $x = 0.0,
    public float $y = 0.0,
    public float $z = 0.0,
  ) {}
}
```

Union Types

Following representation shows the example of native union type declarations in PHP 7.0 and PHP 8.0 respectively:

PHP 7.0

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

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

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

PHP 8.0

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

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

Match Expression

Following representation shows the example of switch v/s match case in PHP 7.0 and PHP 8.0 respectively:

PHP 7.0

```
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 8.0

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

Nullsafe Operator

Following representation shows the example of the use of nullsafe operator ?– in PHP 7.0 and PHP 8.0:

PHP 7.0

```
$country = null;

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

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

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

PHP 8.0

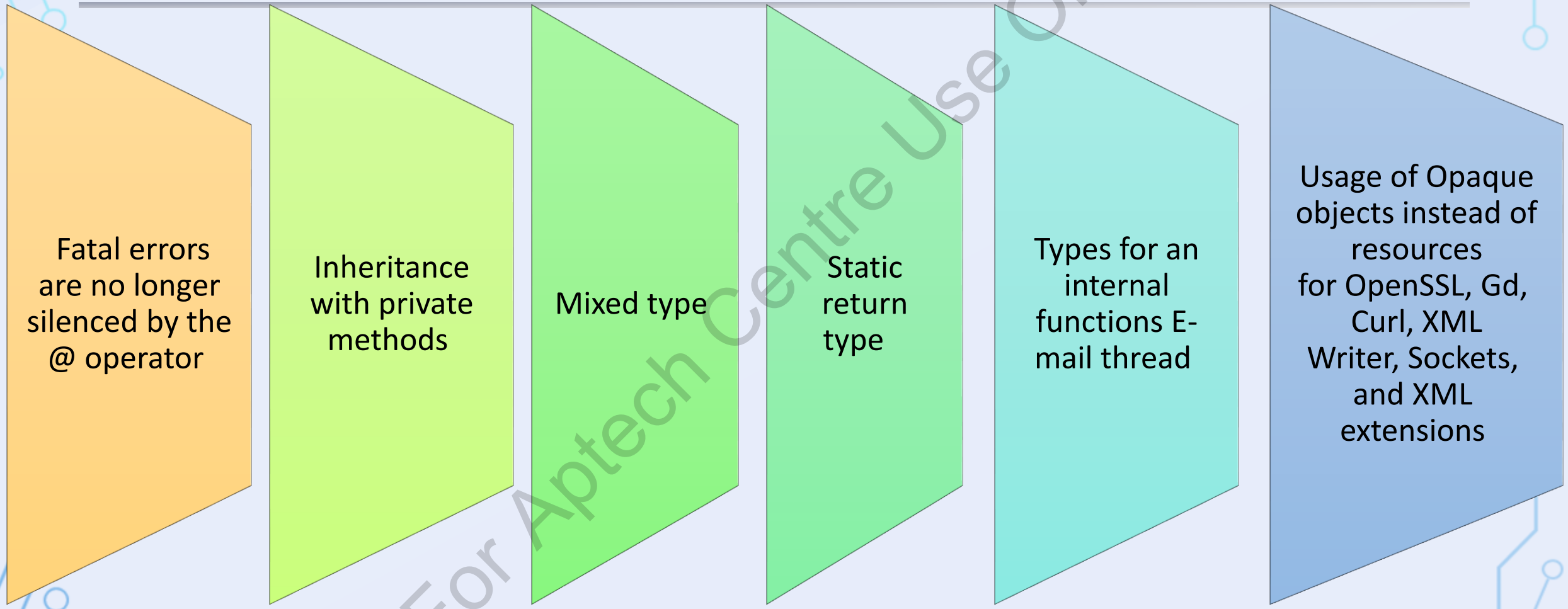
```
$country = $session?->user?->getAddress()?->country;
```

Enhanced Error Handling and Type System [1-2]

Enhancements and modifications in different system types and error handling improvements:

- Stricter type checks for arithmetic/bitwise operators as compared to earlier versions
- Validating abstract trait method
- Magic methods now have correct signatures
- Engine warnings have been reclassified
- Incompatible method signatures give fatal errors

Enhanced Error Handling and Type System [2-2]



Consistency

PHP 8.0 has consistent type errors for internal functions. Most of the internal functions throw an Error exception when there is a failure of validation of the parameters.

Just In Time (JIT) Compiler [1-2]

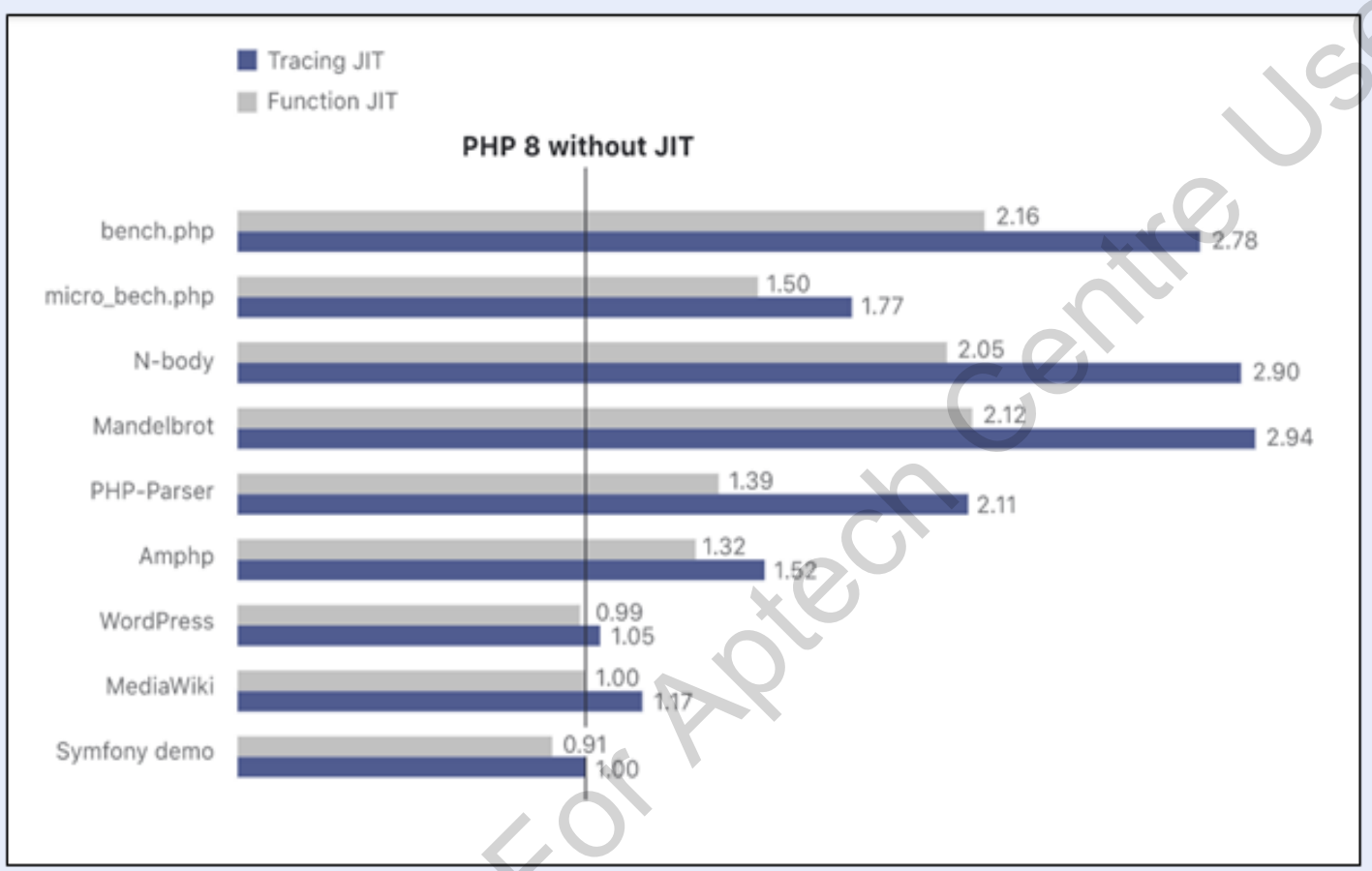
Being a deciphered or an interpreted language, PHP on compilation, does not run immediately at launch time. It runs in real-time.

In earlier versions of PHP, during compiling, whenever a PHP code is run, the interpreter must first interpret, then compile, and finally execute the code.

This is done repeatedly for each request, resulting in slow code execution and wastage of CPU resources.

This problem has now been overcome with the introduction of the Just In Time (JIT) compiler, in PHP 8.0.

Just In Time (JIT) Compiler [2-2]



Relative JIT Contribution to PHP 8.0 Performance

Summary

- PHP is a widely-used open source general-purpose scripting language, especially suited for Web development.
- PHP is easy to learn for freshers and includes several advanced features for seasoned developers.
- Users such as Website developers, WordPress plugin, and theme creators must upgrade their knowledge of PHP to stay ahead of their competitors.
- PHP 8.0 is considered as a major update of the PHP language, with many new features and optimizations.
- Just In Time (JIT) compiler is the most significant feature added in PHP 8.0.
- PHP can dynamically generate content of types such as HTML, Flash, PDF, Text, XML, CSV, and many others.
- PHP has useful text processing features, including the Perl Compatible Regular Expressions (PCRE).