01 – Introduction to PHP

CET218 – Advanced Web Programming

Dr. Ahmed Said

Course Outline

BackEnd

1. Introduction
2. PHP
3. PHP files
4. PHP forms
5. OO PHP

FrontEnd

1. JavaScript
2. VueJs

Grading (100%)

- 20% Homework
- 20% Quizzes
- 20% Project
- Midterm (15%)
- Final (25%)

HTTP and HTML

- HTTP is a communication standard governing the requests and responses that are sent between the browser running on the end user's computer and the web server.
- The server's job is to accept a request from the client and attempt to reply to it in a meaningful way, usually by serving up a requested web page—that's why the term server is used.
- The natural counterpart to a server is a client, so that term is applied both to the web browser and the computer on which it's running.
- Between the client and the server there can be several other devices, such as routers, proxies, gateways, and so on.

HTTP and HTML

- Some of these in-between devices can also help speed up the internet by storing pages or information locally in what is called a cache and then serving this content up to clients directly from the cache rather than fetching it all the way from the source server.
- A web server can usually handle multiple simultaneous connections, and when not communicating with a client, it spends its time listening for an incoming connection. When one arrives, the server sends back a response to confirm its receipt.

The Request/Response Procedure

http://server/path/file

- Usually when you type a URL in your browser:
  - Your computer looks up the server's IP address using DNS
  - Your browser connects to that IP address and requests the given file
  - The web server software (e.g. Apache) grabs that file from the server's local file system, and sends back its contents to you

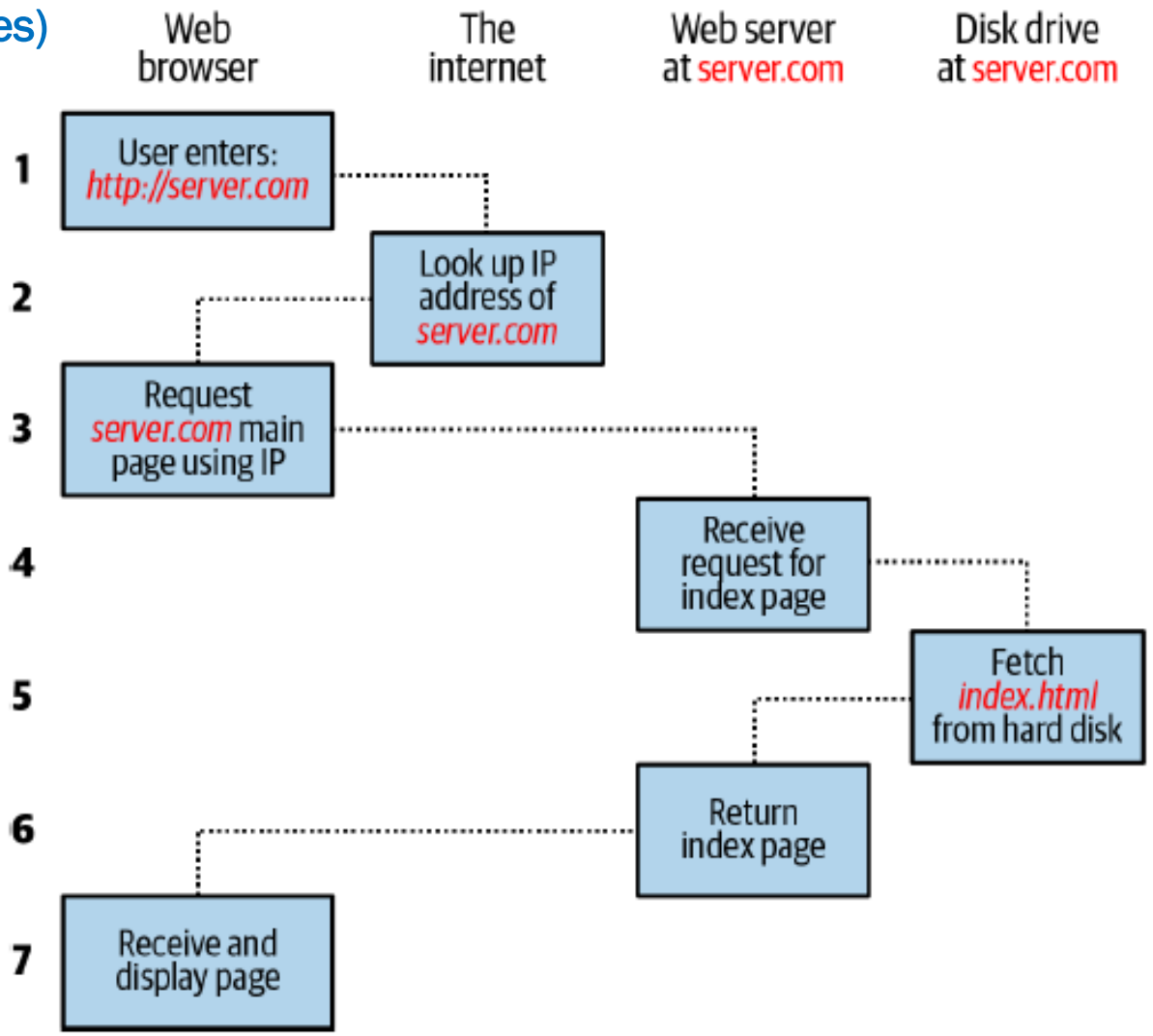
- Some URLs actually specify programs that the web server should run, and then send their output back to you as the result:

https://server.com/quote.php

  - The above URL tells the server server.com to run the program *quote.php* and send back its output

The Request/Response Procedure

Static Content (Web Pages)

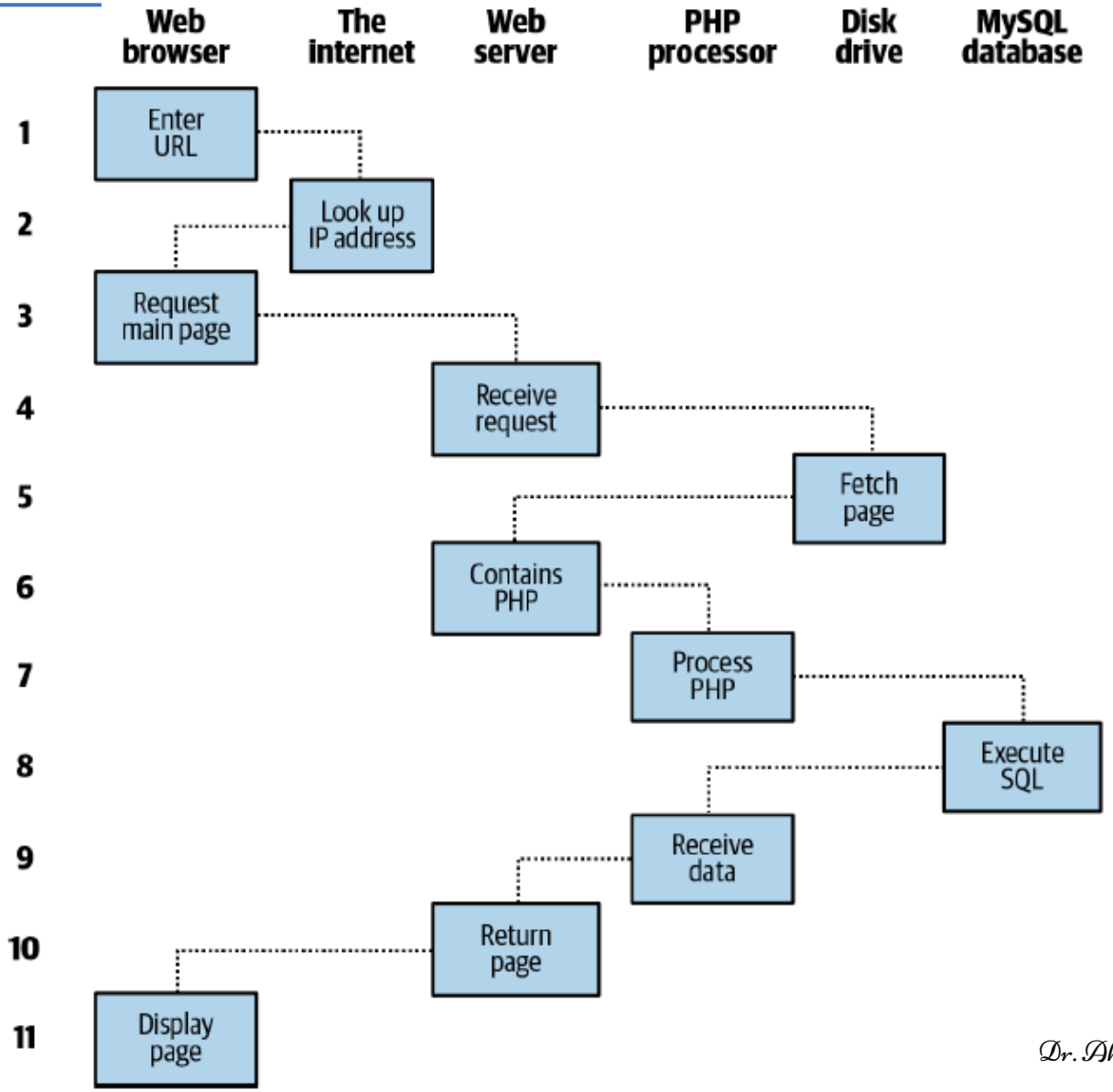


The Request/Response Procedure

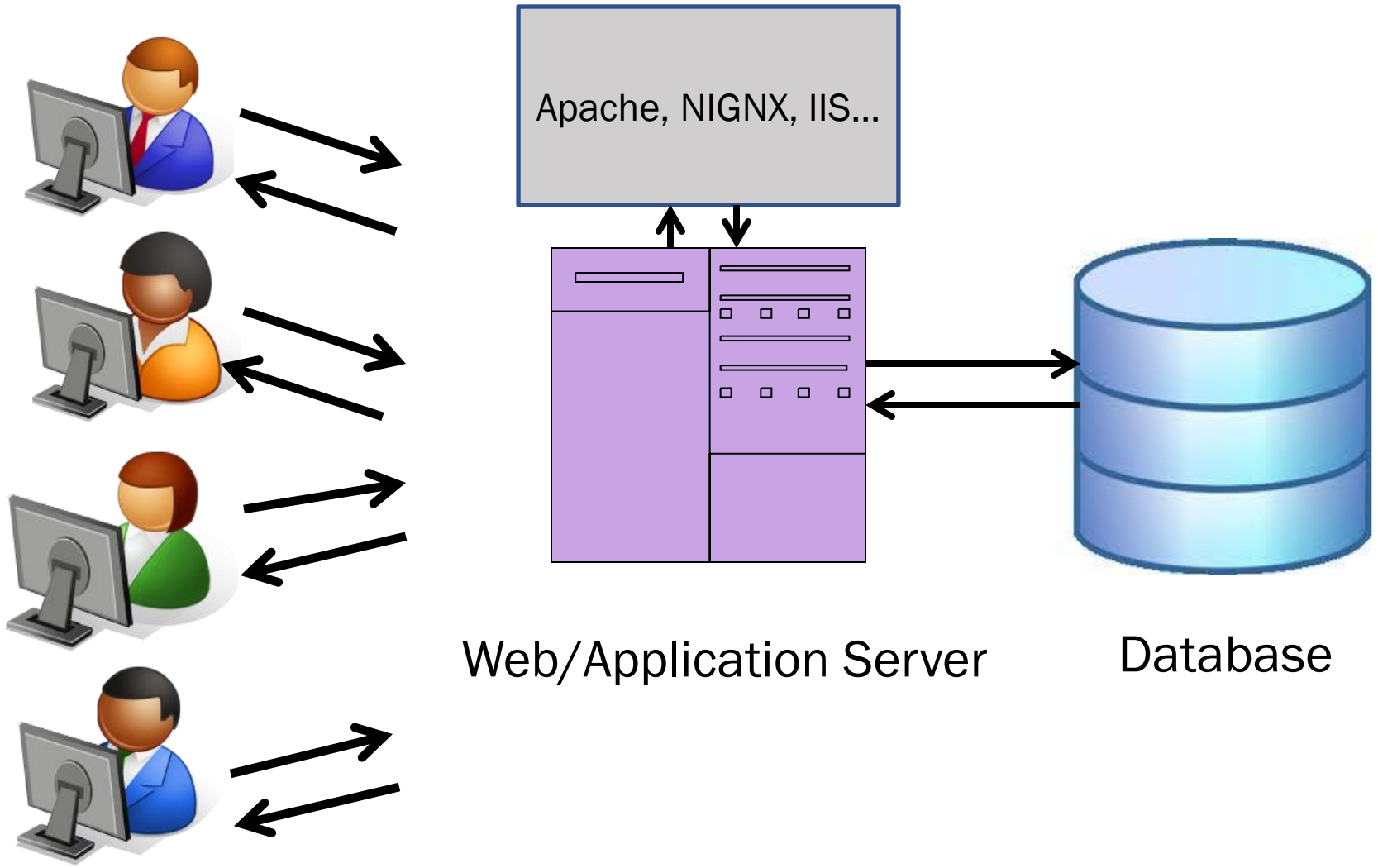
- **Dynamic web pages ?**
- The procedure is a little more involved, because it may bring both
  - Server-Side Programming Language such as PHP, ASP.NET, … and
  - Database Server such as MySQL, … into the mix.
- For instance, you may click a picture of a raincoat.
- Then PHP will put together a request using the standard database language,
- SQL—many of whose commands you will learn in this book—and send the request to
- the MySQL server. The MySQL server will return information about the raincoat you
- selected, and the PHP code will wrap it all up in some HTML, which the server will
- send to your browser

The Request/Response Procedure

- Dynamic web pages ?



The Request/Response Procedure



Web Technologies

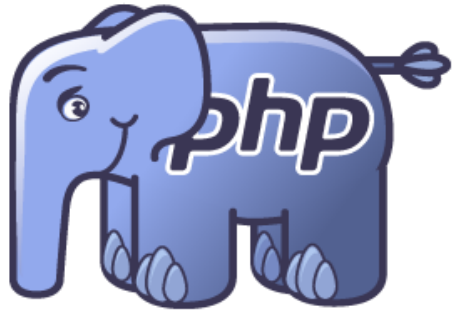
- Client Side
  - HTML5
  - CSS
  - JavaScript
- Server Side
  - Web Server
  - Server-side Programming Language
  - Database Server

Web Server

- Contains software that allows it to run Server-side programs/scripts
- Sends back their output as responses to web requests
  - Apache
  - NIGNX
  - IIS
  - …

Server-Side web programming

- Server-side pages are programs written using one of many web programming languages/frameworks
  - PHP
  - Java/JSP/Spring
  - Ruby on Rails
  - ASP.NET
  - Python
  - Perl
  - Go
  - JavaScript
  - …



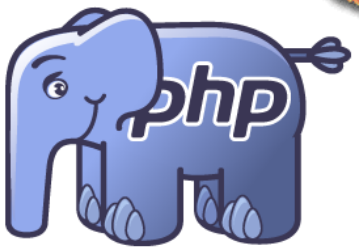
Database

- There are several databases that are widely used on the web, each with its strengths and suited to different types of projects.
  - MySQL
  - MariaDB
  - PostgreSQL
  - SQLite
  - Firebase
  - …

After Oracle purchased Sun Microsystems (the owners of MySQL), the community became wary that MySQL might not remain fully open source, so MariaDB was forked from it to keep it free under the GNU GPL.

Our Stack

- The real beauty of PHP, MySQL, JavaScript (sometimes aided by React, Vue or other frameworks), CSS, and HTML5 is the wonderful way in which they all work together to produce dynamic web content:
  - PHP handles all the main work on the web server
  - MySQL manages all the data
  - The combination of CSS and JavaScript looks after web page presentation.
- Client Side (Frontend)
  - HTML5
  - CSS
  - JavaScript
  - Vue Js
- Server-side (Backend)
  - Apache
  - PHP
  - MySQL/MariaDB

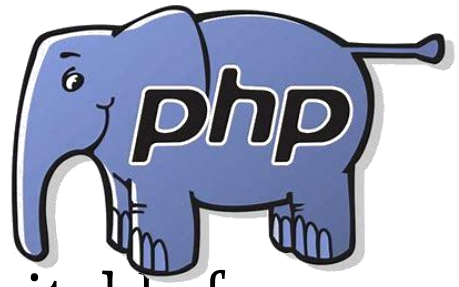


Setting Up a Development Server

- **WAMP**, **MAMP**, and **LAMP** are abbreviations for "Windows, Apache, MySQL, and PHP," "Mac, Apache, MySQL, and PHP," and "Linux, Apache, MySQL, and PHP."
- WAMPs, MAMPs, and LAMPs come in the form of packages that bind the bundled programs together so that you don't have to install and set them up separately.
- Text Editor / IDE
  - VSCode
  - PHPStorm
  - Notepad / Notepad++
  - …

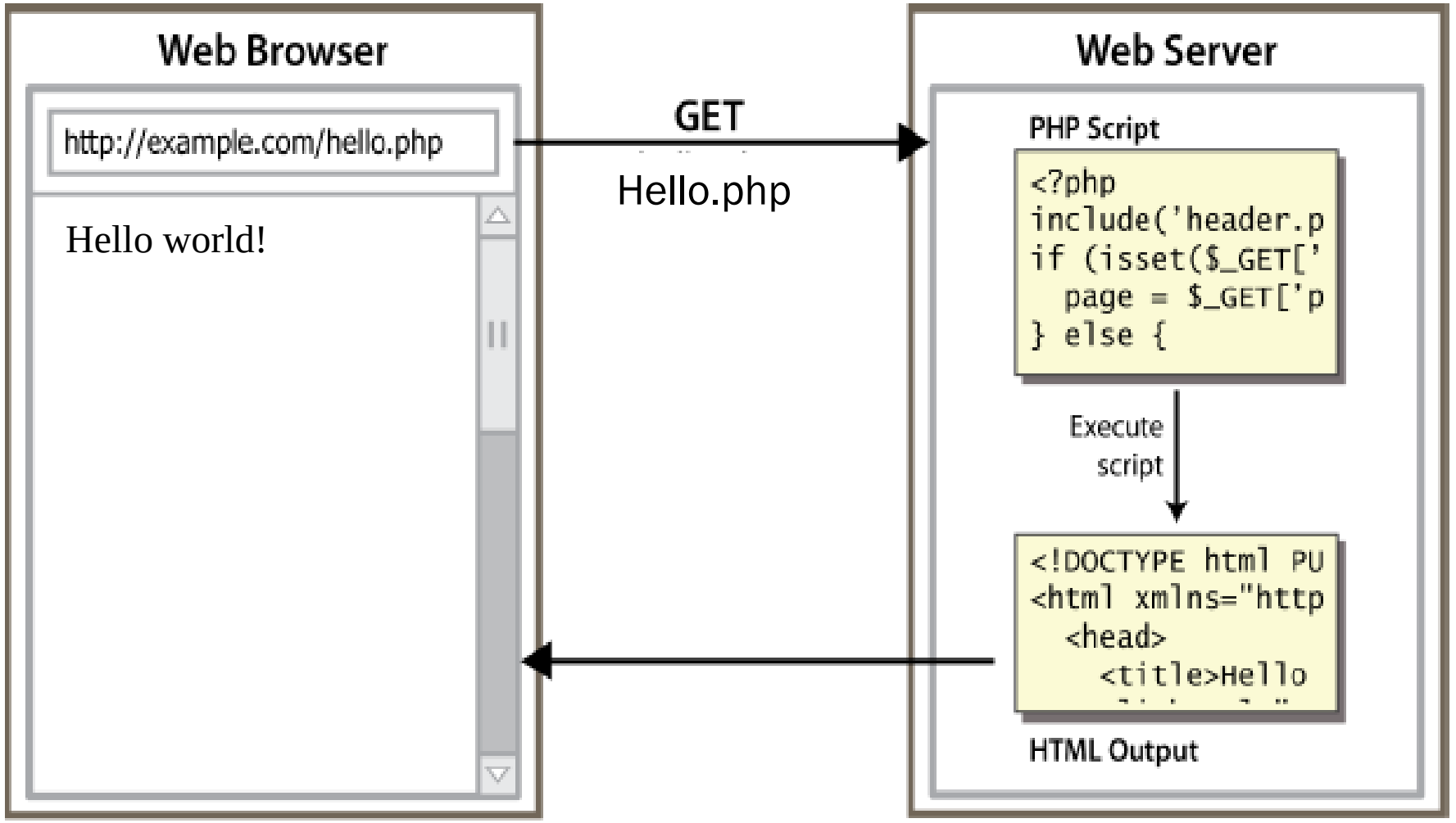
What is PHP?

- **PHP** stands for "Hypertext Preprocessor“ Initially, it was known as "**P**ersonal **H**ome **P**age" when it was created by Rasmus Lerdorf in 1994.
- Used to make web pages dynamic:
  - Provide different content depending on context
  - Interface with other services: database, e-mail, etc.
  - Authenticate users
  - Process form information



- Your PHP program is responsible for passing back a clean file suitable for display in a web browser. At its very simplest, a PHP document will output only HTML.

Lifecycle of a PHP web request



User's computer

Server computer

Why PHP?

- **Free and open source**: anyone can run a PHP-enabled server free of charge
- **Simple**: lots of built-in functionality; familiar syntax
- **Available**: installed on most commercial web servers/hosts
- **Popular / Compatible**
  - Supported by most popular web servers
  - It has since evolved into a widely-used server-side scripting language for web development (used by approximately **74.9%** of all websites in 2025) according to w3
- **Well-documented**: type *php.net/functionName* in browser Address bar to get docs for any function

PHP Basic Syntax

PHP syntax

- To trigger the PHP commands, you need to learn a new tag.
  - **<?php … ?>**
- Contents of a .php file between **<?php** and **?>** are executed as PHP code
- All other contents are output as pure HTML
- We can switch back and forth between HTML and PHP "modes"

```
HTML content
<?php
PHP code
?>
HTML content
<?php
PHP code
?>
HTML content ...
```

PHP

Hello World!

```php
<?php
    echo "Hello world!";
?>
```

PHP

```
Hello world!
```

output

Hello World!

- By the way, there is a slight variation to the PHP syntax. If you browse the internet for PHP examples, you may also encounter code where the opening and closing syntax looks like this:

```php
<?
    echo "Hello world!";
?>
```

PHP

```
Hello world!
```

output

- Although it's not as obvious that the PHP parser is being called, this is a valid, alternative syntax that also usually works.
- But I discourage its use, as it is incompatible with XML and is now deprecated (meaning that it is no longer recommended and support could be removed in future versions).

Hello World!

- If you have only PHP code in a file, you may omit the closing ?>.
- This can be a good practice, as it will ensure that you have no excess whitespace leaking from your PHP files (especially important when you're writing object-oriented code).

```php
<?php
    echo "Hello world!";
```

PHP

```
Hello world!
```

output

Comments

```php
# single-line comment

// single-line comment

/*
multi-line comment
*/
```

- Like C, C++, C# & Java, but # is also allowed

Basic Syntax

- **PHP** is quite a simple language with roots in **C** and **Perl** (if you have ever come across these), yet it looks more like **Java**.
- It is also very flexible, but there are a few rules that you need to learn about its syntax and structure.
- **Semicolons**
  - You may have noticed in the previous examples that the PHP commands ended with
  - a semicolon, like this:

```
$x += 10;
```

Variables

- In PHP, however, you must place a $ in front of all variables. This is required to make the PHP parser faster, as it instantly knows whenever it comes across a variable.
- Names are case sensitive
- Names always begin with $, on both declaration and usage
- Always implicitly declared by assignment (type is not written)
- A loosely typed language (like JavaScript or Python)

```
$mycounter = 1;
$mystring = "Hello";
```

- Unlike languages such as Python, which are very strict about how you indent and lay out your code, PHP leaves you completely free to use (or not use) all the indenting and spacing you like.

Variables

- String

```
$mystring = "Hello";
```

- Numeric (Integers, Floating Point, …)

```
$x = 1;
$x = 17.5;
```

- Arrays
  - The first element of a PHP array is the zeroth element

```
$myarray = array("One", "Two", "Three");
echo $myarray[0];
```

- Two-dimensional arrays

```
$oxo = array(   array('x', ' ', 'o'),
                array('o', 'o', 'x'),
                array('x', 'o', ' ')    );
echo $oxo[1][2];
```

Variables

- Variable-naming rules
  - Variable names, after the dollar sign, must start with a letter of the alphabet or the _ (underscore) character.
  - Variable names can contain only the characters a-z, A-Z, 0-9, and _ (underscore).
  - Variable names may not contain spaces. If a variable name must comprise more than one word, a good idea is to separate the words with the _ (underscore) character (e.g., $user_name).
  - Variable names are case-sensitive. The variable $High_Score is not the same as the variable $high_score.

Operators

- Operators let you specify mathematical operations to perform, such as addition, subtraction, multiplication, and division.
- But several other types of operators exist too, such as the string, comparison, and logical operators.
- **Arithmetic operators**

Operator	Description	Example
+	Addition	`$j + 1`
-	Subtraction	`$j - 6`
*	Multiplication	`$j * 11`
/	Division	`$j / 4`
%	Modulus (the remainder after a division is performed)	`$j % 9`
++	Increment	`++$j`
--	Decrement	`--$j`
**	Exponentiation (or power)	`$j**2`

Operators

- **Assignment operators**
  - These operators assign values to variables. They start with the very simple = and move on to +=, -=, and so on

Operator	Example	Equivalent to
=	$j = 15	$j = 15
+=	$j += 5	$j = $j + 5
-=	$j -= 3	$j = $j - 3
*=	$j *= 8	$j = $j * 8
/=	$j /= 16	$j = $j / 16
.=	$j .= $k	$j = $j . $k
%=	$j %= 4	$j = $j % 4

Operators

- **Comparison operators**
  - Comparison operators are generally used inside a construct such as an if statement in which you need to compare two items.
  - For example, you may wish to know whether a variable you have been incrementing has reached a specific value, or whether another variable is less than a set value, and so on

Operator	Description	Example
==	Is *equal* to	`$j == 4`
!=	Is *not equal* to	`$j != 21`
>	Is *greater than*	`$j > 3`
<	Is *less than*	`$j < 100`
>=	Is *greater than or equal* to	`$j >= 15`
<=	Is *less than or equal* to	`$j <= 8`
<>	Is *not equal* to	`$j <> 23`
===	Is *identical* to	`$j === "987"`
!==	Is *not identical* to	`$j !== "1.2e3"`

Operators

- **Logical operators**
  - you generally use a logical operator to combine the results of two of the comparison operators

Operator	Description	Example
&&	*And*	`$j == 3 && $k == 2`
and	Low-precedence *and*	`$j == 3 and $k == 2`
||	*Or*	`$j < 5 || $j > 10`
or	Low-precedence *or*	`$j < 5 or $j > 10`
!	*Not*	`! ($j == $k)`
xor	*Exclusive or*	`$j xor $k`

  - Note that `&&` is usually interchangeable with and; the same is true for `||` and or. However, because `and` and `or` have a lower precedence, you should avoid using them

Variable Typing

- PHP is a loosely typed language..
- For example, you can create a multiple-digit number and extract the nth digit from it simply by assuming it to be a string.

```php
<?php
    $number = 12345 * 67890;
    echo substr($number, 3, 1);
?>
```

- The numbers `12345` and `67890` are multiplied together, returning a result of `838102050`, which is then placed in the variable `$number`.
- PHP turns `$number` into a nine-character string so that `substr` can access it and return the character, which in this case is `1`.

Variable Typing

```php
<?php
    $pi = "3.1415927";
    $radius = 5;
    echo $pi * ($radius * $radius);
?>
```

- The variable $pi is set to a string value, which is then automatically turned into a floating-point number in the third line by the equation for calculating a circle's area, which outputs the value 78.5398175.

Variable Typing

- Output?

```php
<?php
    $x = "3f";
    echo $x + 2;

    $x = "3.5f";
    echo $x + 2;

    $x = "f3";
    echo $x + 2;
?>
```

for loop

```
for (initialization; condition; update) {
        statements;
}
```

PHP

```
for ($i = 0; $i < 10; $i++) {
    print "$i squared is " . $i * $i . ".\n";
}
```

PHP

if/else statement

```php
if (condition) {
  statements;
} else if (condition) {
  statements;
} else {
  statements;
}
```

PHP

- can also say elseif instead of else if

while loop (same as Java)

```php
while (condition) {
  statements;
}
```

```php
do {
  statements;
} while (condition);
```

- break and continue keywords also behave as in C Family Languages

Console output: print

```php
print "text";
```

PHP

```php
print "Hello, World!\n";
print "Escape \"chars\" are the SAME as in Java!\n";
print "You can have
line breaks in a string.";
print 'A string can use "single-quotes". It\'s cool!';
```

PHP

```
Hello world!  Escape "chars" are the SAME as in Java! You can have line
breaks in a string. A string can use "single-quotes". It's cool!
```

output

echo vs print ?

THANK YOU