

Дисципліна

«Бекенд-розробка, сервіси та мікросервіси»

КОНСПЕКТ ЛЕКЦІЇ. ТЕМА 2

Змінні, області видимості, суперглобальні змінні, константи, типи даних, оператори, передача за посиланням, дата й час, робота з рядками в PHP

Спеціальність: Комп'ютерна інженерія, 3 курс
Обсяг: 1 год 20 хв (лекція 2 з 18)

План лекції

1. Змінні в PHP: оголошення та іменування
2. Області видимості змінних
3. Суперглобальні змінні
4. Вивід значень змінних на екран
5. Константи
6. Магічні константи
7. Коментарі
8. Типи змінних
9. Оператори в PHP
10. Передача значень за посиланням
11. Робота з датою та часом
12. Рядкові величини: форматування та парсинг

Мета лекції: опанувати базовий інструментарій PHP для роботи з даними — від оголошення змінної до форматування рядків і дат, — щоб надалі впевнено писати логіку серверних скриптів.

1. Змінні в PHP

1.1. Оголошення та іменування

Змінна в PHP — це іменована область пам'яті для зберігання даних. Кожна змінна починається символом долара \$, за яким слідує ім'я. На відміну від багатьох мов, PHP не вимагає заздалегідь оголошувати тип змінної: тип визначається автоматично на основі присвоєного значення (динамічна типізація), а сама змінна створюється в момент першого присвоєння.

```
<?php
$name = "Мар'ян";      // рядок (string)
$age = 21;             // ціле число (int)
$price = 19.99;        // число з плаваючою крапкою (float)
$isActive = true;      // логічне значення (bool)

// Змінна може змінити тип у будь-який момент
$value = 10;           // int
$value = "десять";     // тепер string
```

1.2. Правила іменування

- Ім'я змінної починається з літери або символу підкреслення `_`, далі можуть йти літери, цифри та підкреслення (регулярний вираз `[a-zA-Z_][a-zA-Z0-9_]*`).
- Імена змінних є регістрозалежними: `$name` і `$Name` — дві різні змінні.
- Ім'я не може починатися з цифри: `$1value` — синтаксична помилка.
- Прийнята конвенція іменування — `camelCase` (`$totalPrice`) або `snake_case` (`$total_price`); головне — послідовність у межах проєкту.

PHP підтримує також змінні змінні (variable variables) — `$$name`, коли ім'я змінної саме зберігається в іншій змінній. Це рідкісна техніка, яку варто знати, але майже ніколи не використовувати на практиці через зниження читабельності коду.

2. Області видимості змінних

Область видимості (scope) визначає, у якій частині програми змінна доступна для використання. У PHP, на відміну від багатьох мов, кожна функція має власну ізольовану область видимості — змінні, оголошені поза функцією, автоматично не потрапляють усередину функції.

2.1. Локальна область видимості

```
<?php
$counter = 10;    // глобальна змінна

function increment() {
    $counter = 1;  // це ІНША, локальна змінна
    $counter++;
    echo $counter; // виведе 2
}

increment();
echo $counter;    // виведе 10 — глобальна змінна не змінилась
```

2.2. Ключове слово global

Щоб функція могла звертатися до глобальної змінної напямую, потрібно явно оголосити її ключовим словом `global` усередині функції, або звернутися через суперглобальний масив `$GLOBALS` (розділ 3).

```
<?php
$counter = 10;

function increment() {
    global $counter;
    $counter++;
}

increment();
echo $counter; // виведе 11
```

2.3. Статичні змінні

Ключове слово `static` зберігає значення локальної змінної між викликами функції — на відміну від звичайної локальної змінної, яка «обнуляється» щоразу під час нового виклику.

```
<?php
function countCalls() {
    static $calls = 0;
    $calls++;
    echo "Викликів: $calls\n";
}

countCalls(); // Викликів: 1
countCalls(); // Викликів: 2
countCalls(); // Викликів: 3
```

Статичні змінні часто використовують для лічильників, кешування результату всередині функції або для реалізації простих одноразових ініціалізацій без використання глобального стану.

3. Суперглобальні змінні

Суперглобальні змінні (superglobals) — це вбудовані масиви PHP, які автоматично доступні в будь-якій області видимості (у тому числі всередині функцій) без потреби оголошувати `global`. Вони є основним способом отримати дані з HTTP-запиту, середовища сервера чи сесії користувача.

Суперглобальна	Призначення
<code>\$_GET</code>	Параметри рядка запиту (query string), напр. <code>?id=5</code>
<code>\$_POST</code>	Дані, надіслані методом POST (форми)
<code>\$_REQUEST</code>	Об'єднання <code>\$_GET</code> , <code>\$_POST</code> і <code>\$_COOKIE</code>
<code>\$_SERVER</code>	Інформація про сервер і середовище виконання
<code>\$_SESSION</code>	Дані користувацької сесії (після <code>session_start()</code>)
<code>\$_COOKIE</code>	HTTP-куки, надіслані клієнтом
<code>\$_FILES</code>	Інформація про завантажені файли
<code>\$_ENV</code>	Змінні середовища операційної системи
<code>\$GLOBALS</code>	Доступ до всіх глобальних змінних із будь-якої області

```
<?php
// URL: http://localhost/profile.php?id=42
$id = $_GET['id'];           // "42"

// Форма з методом POST
$email = $_POST['email'] ?? ''; // порожній рядок, якщо не надіслано

// Інформація про сервер
echo $_SERVER['REQUEST_METHOD']; // GET або POST
echo $_SERVER['HTTP_HOST'];      // shop.example.com
```

Дані з `$_GET`, `$_POST` і будь-яких інших зовнішніх джерел ніколи не можна вважати безпечними — їх завжди потрібно валідувати й екранувати перед використанням (у SQL-запитах, виводі в HTML тощо). Ця тема детально розглядається в наступних лекціях курсу.

4. Вивід значень змінних на екран

4.1. echo та print

echo та print — мовні конструкції (а не функції) для виводу даних у потік HTTP-відповіді. echo дозволяє передавати кілька значень через кому й трохи швидша; print завжди повертає значення 1, тому її можна використовувати у виразах.

```
<?php
echo "Привіт, світе!";
echo "Ім'я: ", $name, ", вік: ", $age;    // кілька аргументів
print "Це теж виведе рядок";
```

4.2. print_r, var_dump, var_export

Для налагодження (debugging) складних структур даних (масивів, об'єктів) звичайного echo недостатньо — він не вміє виводити масиви напряму. Для цього PHP надає спеціальні функції:

Функція	Призначення
print_r()	Виводить структуру масиву/об'єкта в читабельному вигляді
var_dump()	Виводить значення разом із типом даних і розміром
var_export()	Виводить значення у вигляді коду, придатного для повторного виконання

```
<?php
$user = ["name" => "Оксана", "age" => 25];

print_r($user);
// Array ( [name] => Оксана [age] => 25 )

var_dump($user);
// array(2) { ["name"]=> string(12) "Оксана" ["age"]=> int(25) }
```

На практиці var_dump() — найкорисніший інструмент під час налагодження: він показує не лише значення, а й точний тип даних, що часто допомагає знайти помилки, пов'язані з неявним перетворенням типів.

5. Константи

Константа — це ідентифікатор для значення, яке не може бути змінене після визначення. На відміну від змінних, константи не мають префікса \$ і за конвенцією записуються великими літерами. Визначити константу можна двома способами: функцією `define()` або ключовим словом `const`.

```
<?php
define('SITE_NAME', 'IFNTUOG Portal');
define('MAX_UPLOAD_SIZE', 5 * 1024 * 1024); // 5 МБ

const VERSION = '1.0.0';

echo SITE_NAME;    // без символу $
echo VERSION;
```

Критерій	<code>define()</code>	<code>const</code>
Область застосування	Будь-де, у тому числі умовно	Лише на верхньому рівні файлу або в класі
Час визначення	Під час виконання (runtime)	Під час компіляції
Використання в класах	Не застосовується для властивостей класу	Використовується для констант класу

6. Магічні константи

Магічні константи — це спеціальні вбудовані константи PHP, значення яких залежить від контексту, у якому вони використовуються (файл, рядок, функція, клас). Їх назви завжди обрамлені подвійними символами підкреслення.

Константа	Повертає
__LINE__	Поточний номер рядка у файлі
__FILE__	Повний шлях і ім'я поточного файлу
__DIR__	Директорію поточного файлу
__FUNCTION__	Ім'я поточної функції
__CLASS__	Ім'я поточного класу
__METHOD__	Ім'я поточного методу класу (Клас::метод)
__NAMESPACE__	Ім'я поточного простору імен

```
<?php
function logError($message) {
    echo "[" . __FUNCTION__ . "] Рядок " . __LINE__ . ": $message";
}

logError("Файл не знайдено");
// [logError] Рядок 5: Файл не знайдено
```

Магічні константи особливо корисні для логування помилок і налагодження — вони автоматично показують, де саме в коді відбулася подія, без потреби вручну прописувати ці дані.

7. Коментарі

Коментарі — це фрагменти коду, які інтерпретатор ігнорує під час виконання. Вони використовуються для пояснення логіки, тимчасового вимкнення частини коду та документування функцій і класів. PHP підтримує три стилі коментарів:

```
<?php
// Однорядковий коментар (стиль C++)
# Однорядковий коментар (стиль Perl/shell)

/*
 * Багаторядковий коментар.
 * Часто використовується для блоків документації.
 */

/**
 * PHPDoc-коментар — використовується IDE для підказок
 * @param string $name Ім'я користувача
 * @return string Привітання
 */
function greet($name) {
    return "Привіт, $name!";
}
```

*Стиль PHPDoc (/** ... */ з тегами @param, @return тощо) — фактичний стандарт індустрії: сучасні IDE (PhpStorm, VS Code) використовують ці коментарі для автодоповнення й перевірки типів, навіть якщо сам PHP їх ігнорує під час виконання.*

8. Типи змінних

8.1. Скалярні типи

Тип	Опис	Приклад
int	Ціле число	<code>\$age = 21;</code>
float (double)	Число з плаваючою крапкою	<code>\$price = 19.99;</code>
string	Рядок символів	<code>\$name = "Мар'ян";</code>
bool	Логічне значення true/false	<code>\$isActive = true;</code>

8.2. Складені та спеціальні типи

Тип	Опис
array	Впорядкована або асоціативна колекція значень
object	Екземпляр класу
callable	Значення, яке можна викликати як функцію
null	Спеціальне значення «відсутність значення»
resource	Посилання на зовнішній ресурс (файл, з'єднання з БД)

8.3. Перевірка та перетворення типів

PHP — мова зі слабкою типізацією: значення автоматично приводяться до потрібного типу залежно від контексту (type juggling). Наприклад, рядок "5" при арифметичній операції автоматично перетвориться на число. Для явного контролю типів використовують функції `is_*`() та приведення типів (type casting).

```
<?php
$value = "42";
var_dump(is_string($value)); // bool(true)
var_dump(is_numeric($value)); // bool(true) — рядок містить число

$number = (int) $value;      // явне приведення до int
var_dump($number);          // int(42)

echo gettype($number);      // "integer"
settype($number, "string"); // змінює тип змінної на місці
```

Із версії PHP 8 доступний суворіший режим — `declare(strict_types=1)` на початку файлу — який вимикає автоматичне приведення типів для аргументів функцій із оголошеними типами. Це наближає PHP до практик статично типізованих мов і рекомендується для нових проєктів.

9. Оператори в PHP

9.1. Арифметичні та рядкові оператори

Оператор	Дія	Приклад
+ - * /	Додавання, віднімання, множення, ділення	5 + 3 → 8
%	Остача від ділення	10 % 3 → 1
**	Піднесення до степеня	2 ** 8 → 256
.	Конкатенація (з'єднання) рядків	"a" . "b" → "ab"
++ --	Інкремент / декремент	\$i++

9.2. Оператори порівняння

Оператор	Значення
==	Рівність значень (з приведенням типів)
===	Строго рівність (значення й тип)
!= <>	Нерівність значень
!==	Строго нерівність
< > <= >=	Менше, більше, менше або дорівнює, більше або дорівнює
<=>	Оператор "космічний корабель": повертає -1, 0 або 1

Найпоширеніша помилка початківців — використання == замість ===. Наприклад, "0" == false і навіть "abc" == 0 у старих версіях PHP повертали true через неявне приведення типів. У сучасному коді майже завжди варто використовувати строге порівняння ===.

9.3. Логічні та інші важливі оператори

Оператор	Значення
&& and	Логічне І
or	Логічне АБО
!	Логічне заперечення (НЕ)
?:	Скорочений тернарний оператор (Elvis)
??	Оператор об'єднання з null (null coalescing)
??=	Присвоєння, якщо змінна ще не визначена або дорівнює null

```
<?php
$username = $_GET['user'] ?? 'Гість';    // якщо ключа немає – 'Гість'

$count = $count ?? 0;
$count ??= 0;    // те саме, коротший запис

$label = ($age >= 18) ? 'дорослий' : 'неповнолітній';
```

10. Передача значень за посиланням

За замовчуванням PHP передає аргументи у функції за значенням (by value) — усередині функції створюється копія даних, і зміни не впливають на оригінальну змінну. Щоб функція могла змінювати оригінальну змінну, використовують передачу за посиланням (by reference) — символ & перед ім'ям параметра.

```
<?php
// Передача за значенням (за замовчуванням)
function addTax($price) {
    $price = $price * 1.2;
    return $price;
}
$original = 100;
$withTax = addTax($original);
echo $original;    // 100 — не змінилось
echo $withTax;     // 120

// Передача за посиланням
function addTaxByRef(&$price) {
    $price = $price * 1.2;
}
$original = 100;
addTaxByRef($original);
echo $original;    // 120 — змінилось напряму!
```

Посилання можна створювати й між звичайними змінними за допомогою оператора =&: обидві змінні тоді вказують на ту саму область пам'яті, і зміна однієї з них автоматично відображається в іншій.

```
<?php
$a = 5;
$b = &$a;    // $b тепер посилання на $a
$b = 10;
echo $a;     // 10 — обидві змінні "дивляться" на одне значення
```

Передача за посиланням часто використовується у функціях сортування масивів (наприклад, вбудована `sort($array)` модифікує масив напряму) та коли функція має повертати кілька значень без створення додаткового масиву.

11. Робота з датою та часом

11.1. Функції date() і time()

Функція time() повертає поточний Unix-час — кількість секунд, що минули з 1 січня 1970 року (00:00:00 UTC), так звана «епоха Unix». Функція date() форматує часову мітку у зручний для читання рядок відповідно до заданого формату.

```
<?php
echo time();                // 1798113600 (приклад)
echo date("Y-m-d");         // 2026-09-02
echo date("d.m.Y H:i:s");   // 02.09.2026 14:30:00
echo date("l, d F Y");      // Wednesday, 02 September 2026
```

Символ	Значення	Символ	Значення
Y	Рік, 4 цифри (2026)	H	Година, 24-год формат
y	Рік, 2 цифри (26)	i	Хвилини
m	Місяць, з нулем (09)	s	Секунди
d	День місяця, з нулем (02)	l	Назва дня тижня
n	Місяць без нуля (9)	F	Повна назва місяця

11.2. Клас DateTime

Об'єктно-орієнтований клас DateTime — сучасніший і гнучкіший спосіб роботи з датами, ніж набір окремих функцій. Він дозволяє легко виконувати арифметику з датами (додавати/віднімати інтервали) та порівнювати дати між собою.

```
<?php
$date = new DateTime('2026-09-02');
$date->modify('+1 month');
echo $date->format('Y-m-d'); // 2026-10-02

$interval = new DateInterval('P7D'); // 7 днів
$date->add($interval);
echo $date->format('Y-m-d'); // 2026-10-09

$deadline = new DateTime('2026-12-31');
$diff = $date->diff($deadline);
echo $diff->days . ' днів залишилось';
```

Функція `strtotime('+1 week')` та конструктор `new DateTime('+1 week')` розуміють природну мову дат англійською ("next monday", "tomorrow", "-3 days") — це зручно для обчислення дедлайнів і термінів дії без ручної арифметики над секундами.

12. Рядкові величини: форматування та парсинг

12.1. Одинарні та подвійні лапки, heredoc

У рядках з одинарними лапками `'...'` PHP не інтерпретує змінні та більшість спецсимволів (окрім `\\` і `\'`). У рядках з подвійними лапками `"..."` змінні автоматично підставляються, а спецсимволи (`\n`, `\t`) інтерпретуються. Для великих багаторядкових шаблонів зручно використовувати синтаксис `heredoc`.

```
<?php
$name = "Оксана";
echo 'Привіт, $name';      // Привіт, $name (без підстановки)
echo "Привіт, $name";      // Привіт, Оксана

$html = <<<EOT
<div>
    <p>Користувач: $name</p>
</div>
EOT;
```

12.2. Форматування: `sprintf` і `printf`

Функції `sprintf()` і `printf()` дозволяють формувати рядки за шаблоном — особливо зручно для вирівнювання чисел, встановлення кількості знаків після коми чи доповнення нулями.

```
<?php
$price = 1234.5;
echo sprintf("Ціна: %.2f грн", $price);      // Ціна: 1234.50 грн
echo sprintf("Номер: %05d", 42);             // Номер: 00042
echo number_format($price, 2, ',', ' ');     // 1 234,50
```

12.3. Основні функції обробки рядків

Функція	Призначення
<code>strlen(\$s)</code>	Довжина рядка в байтах
<code>substr(\$s, \$start, \$len)</code>	Витягнути підрядок
<code>str_replace(\$search, \$repl, \$s)</code>	Замінити всі входження підрядка
<code>trim(\$s)</code>	Видалити пробіли з початку й кінця рядка
<code>strtoupper()</code> / <code>strtolower()</code>	Перетворити регістр символів
<code>explode(\$delim, \$s)</code>	Розбити рядок на масив за роздільником
<code>implode(\$delim, \$arr)</code>	З'єднати масив у рядок через роздільник
<code>str_pad(\$s, \$len, \$ch)</code>	Доповнити рядок символами до потрібної довжини

12.4. Парсинг рядків

Парсинг — це розбір рядка на складові частини за певними правилами. Найпростіший спосіб — `explode()` для розбиття за фіксованим роздільником; для складніших патернів використовують регулярні вирази (функції `preg_match()`, `preg_match_all()`, `preg_replace()`).

```
<?php
$csv = "Іван,25,Львів";
[$name, $age, $city] = explode(',', $csv);
echo "$name живе у місті $city";

// Парсинг за допомогою регулярного виразу
$text = "Телефон: +380671234567";
if (preg_match('/\+380\d{9}/', $text, $matches)) {
    echo "Знайдено номер: " . $matches[0];
}
```

Регулярні вирази (PCRE у PHP) — потужний, але «важкий» інструмент: для простих задач (перевірка наявності підрядка, розбиття за комою) варто спочатку розглянути прості рядкові функції — вони швидші й читабельніші.

Питання для самоперевірки

- Чим локальна область видимості змінної відрізняється від глобальної? Наведіть приклад, де це важливо.
- Для чого призначені статичні змінні (static) і чим вони відрізняються від глобальних?
- Назвіть щонайменше п'ять суперглобальних масивів PHP та їх призначення.
- У чому різниця між echo, print_r() та var_dump()? Коли варто використовувати кожен з них?
- Чим define() відрізняється від const? Наведіть приклад ситуації, де підходить лише один із варіантів.
- Наведіть приклад використання магічної константи __LINE__ або __FUNCTION__ для логування.
- Чому важливо використовувати === замість == під час порівняння значень у PHP?
- Поясніть різницю між передачею аргументу за значенням і за посиланням. Наведіть приклад коду.
- Як отримати дату через 10 днів від поточної за допомогою класу DateTime?
- Чим відрізняється поведінка змінних у рядках з одинарними та подвійними лапками?

Короткий глосарій

Термін	Означення
Область видимості (scope)	Частина програми, у якій змінна доступна для використання
Суперглобальна змінна	Вбудований масив PHP, доступний у будь-якій області видимості
Type juggling	Автоматичне перетворення типу значення залежно від контексту
Магічна константа	Вбудована константа, значення якої залежить від контексту виконання
Передача за посиланням	Спосіб передачі аргументу, за якого функція працює з оригінальними даними
Unix-час	Кількість секунд, що минули з 1 січня 1970 року (00:00:00 UTC)
Парсинг	Розбір рядка на складові частини за певними правилами