



Въведение в PHP

Синтаксис и основи на езика

Атанас Василев
a.vasilev@viscomp.bg
www.viscomp.bg

About us

■ Лектори

- Георги Гроздев – g.grozdev@viscomp.bg
- Атанас Василев – a.vasilev@viscomp.bg

■ Инфо и слайдовете ще намерите на:

- <http://phplab.viscomp.bg/>

vis|comp
we develop the web

Синтаксис

- Файлове. Типични разширения
 - В контекст на веб-сървър:
 - `.php`
 - `.php3`
 - `.phtml`
 - `AddType application/x-httpd-php .php .phtml .php3`
 - Като конзолно приложение (PHP 4.3.0+)
 - `php some_file.php`
 - `php -f some_file.php`
 - `some_file` (ако скриптът съдържа `#!/usr/bin/php`)

Синтаксис

- Излизане от контекста
- Ограждане на блокове от PHP код
 - Стандартни разделители: `<?php ... код ?>`
 - Съкратени разделители: `<? ... код ?>`
 - `<?= $variable ?>` - алиас на `<? echo $variable ?>`
 - Скрипт-разделители: `<script language="php">... код </script>`
 - ASP разделители: `<% ... код %>`
 - Последният затварящ PHP таг (`?>`) в края на файл е незадължителен и понякога се пропуска за да се избегне нежелан output
- Знак за нов ред
 - Първият знак за нов ред, непосредствено след затварящ PHP таг се игнорира от интерпретатора

Синтаксис

- Белите полета (white space) са без значение

Hello world!

```
<html>  
<?php echo "Hello world"; ?>  
</html>
```

■ `echo()` vs. `print()`

- езикови конструкции – не са функции
- `print()` се държи като функция – връща стойност
- Скорост – `echo()` е малко по-бърза
- Параметри
 - `echo "string", "string", "string"; // ok`
 - `echo ("string string string"); // ok`
 - `echo (expression, expression, expression); // fail`
 - `echo (expression), (expression), (expression); // ok`
 - `print()` приема само един параметър

Елементи на езика

- Литерали
- Идентификатори
- Езикови конструкции (резервирани думи)
- Инструкции (statements)
- Коментари
- Групирани инструкции (code blocks)

Елементи на езика

- Литерали
 - Стойности, които се появяват директно в кода
 - 2001
 - 0xFE
 - "Hello world"
 - true
 - NULL

Елементи на езика

- Идентификатори
 - Идентификаторите в PHP са просто имена. С тях се именуват:
 - Променливи
 - `$varname`
 - Функции
 - `do_something()`
 - Константи
 - `CONST_NAME`
 - Класове
 - `class Foo {...}`

Елементи на езика

- Езикови конструкции

- Резервирани думи от PHP, които не могат да се използват за именуване на променливи, константи, функции и класове:

and	or	xor	__FILE__	exception (PHP 5)
__LINE__	array()	as	break	case
class	const	continue	declare	default
die()	do	echo()	else	elseif
empty()	enddeclare	endfor	endforeach	endif
endswitch	endwhile	eval()	exit()	extends
for	foreach	function	global	if
include()	include_once()	isset()	list()	new
print()	require()	require_once()	return()	static
switch	unset()	use	var	while
__FUNCTION__	__CLASS__	__METHOD__	final (PHP 5)	php_user_filter (PHP 5)
interface (PHP 5)	implements (PHP 5)	extends	public (PHP 5)	private (PHP 5)
protected (PHP 5)	abstract (PHP 5)	clone (PHP 5)	try (PHP 5)	catch (PHP 5)
throw (PHP 5)	cfunction (само PHP 4)	old_function (PHP 4 only)	this (само PHP 5)	

Елементи на езика

- Инструкции
 - Извикване на функции
 - Присвоявания на стойности на променливи
 - Извеждане на данни
 - Изрази (expressions) и др.
 - Инструкциите се прекъсват по два начина:
 - точка и запетая - “.”
 - затварящ PHP таг - “?>”
 - Последната инструкция преди затварящ PHP таг не изисква “.”

Елементи на езика

■ Коментари

□ Едноредови коментари

Прекъсват се от `\r`, `\n`, `\r\n`, както и от затварящ PHP таг (`?>`)

```
<?php  
$foo = 1; // Този коментар се прекъсва от края на реда  
$bar = 2; # Този коментар се прекъсва от ?> това ще се отпечата
```

Елементи на езика

■ Коментари

□ Многоредови коментари

```
<?php
$foo = 1;
/* Коментар, който се простира на повече
от един ред
*/

/**
 * Пример за документиране на API
 *
 * @param array $elements
 */
function render_elements($elements) {...}
?>
```

Елементи на езика

- Групирани инструкции (code blocks)
 - Серия от инструкции, оградени с фигурни скоби

```
<?php
{
    foo(); // извикване на функция
    $bar = true; // присвояване на стойност
}
?>
```

- Подходящи за указване на набор от инструкции, които да бъдат изпълнени само при определени условия
- Могат да бъдат вложени

Типове данни

- PHP е слабо-типизиран език
 - Променливите могат да сменят типа на стойностите, които държат за периода на съществуването си
 - Променливите сменят типа на стойностите си в зависимост от контекста, в който се ползват
- Скаларни типове
 - `int`, `float`, `boolean`, `string`
- Съставни типове
 - `array`, `object`
- Специални типове
 - `NULL`, `resource`
- Псевдо-типове
 - `mixed`, `number`, `callback`, `void`

Типове данни

■ `int` (цели числа)

- Цели числа със знак и без знак
- Декларират се чрез няколко различни нотации:
 - **Десетична**: 10; -11; 1452
 - **Осмична**: 0666; 0100 (ползва се главно за указване права за достъп в UNIX-базираните системи)
 - **Шеснайсетична**: 0x123; 0XFF; -0x100 (префиксът 0x не е чувствителен към регистъра - case insensitive)

Типове данни

■ float (числа с плаваща запетая)

- Още doubles, реални числа
- Числа, които имат дробна част
- Също могат да бъдат със знак и без знак
- Декларират се чрез две различни нотации:
 - Десетична: 0.12; 1234.43; -.123
 - Експоненциална: 2E7, 1.2e2
 - Числото се умножава по 10 и се повдига на съответната степен. Например 1e2 е равно на 100
 - “E” не е чувствително към регистъра

Типове данни

■ `boolean` (булеви)

- Съдържат една от две възможни стойности: TRUE или FALSE
- Служат като основа на логически операции (предмет на друга лекция)

■ `string` (низ)

- Подреден списък от двоични знаци
- Не задължително текст - биха могли да представляват графичен или музикален файл
- PHP има изключително много функции за манипулиране на низове

Типове данни

■ array (масив)

- Контейнери с подредени елементи, съдържащи други данни, включително:
 - числа, boolean, string, object, array
- Стойностите в масивите се идентифицират по позиция - или число с база '0', или именован низов идентификатор
- Създаване на масив
 - `$my_array = array('foo', 'bar');`
 - `$my_array[0] = 'foo'; $my_array[1] = 'bar';`
 - `$my_array[] = 'foo'; $my_array[] = 'bar';`
- В PHP има множество функции за проверка дали типа на дадена променлива е array, за получаване броя на елементите в масив, за манипулирането им, за сортиране или търсене между елементите. Предмет на друга лекция

Типове данни

■ object (обект)

- Контейнери едновременно с данни и код
- Формират основата на обектно-ориентираното програмиране

■ resource (ресурсен идентификатор)

- Идентифицират външни ресурси, които принципно не са вградени в PHP, но придобиват значение в контекста на някои специални операции, като например работа с файлове или обработка на картинки

■ NULL

- Показва, че дадена променлива няма стойност
- Променлива има стойност NULL, когато ѝ е била присвоена стойността NULL, или когато изобщо не ѝ е била присвоена стойност

Типове данни

■ `mixed`

- Указва, че параметърът може да бъде променлива от различни (не задължително всички) типове

■ `number`

- Указва, че параметърът може да бъде както `int`, така и `float`

■ `callback`

- Указва, че параметърът е произволна функция, дефинирана от потребителя или метод на обект. В някои случаи се задава като низ, а в други - като масив

■ `void`

- Като тип за връщане (от функция) показва, че върнатата стойност е без значение
- В контекст на параметър показва, че функцията не приема никакви параметри

Променливи

- Държат стойности от някой от изброените типове данни: числа, низове, обекти, масиви, булеви
- Специални контейнери, определени с именован идентификатор, предшестван от знака за долар - '\$'
- Разрешени символи в име на променлива са буквите (A-Z, a-z), цифрите (0-9), както и знака за под-черта ('_')
- Имената са чувствителни към регистъра (case-sensitive)
- Името задължително започва с буква или под-черта

```
$a; // valid
```

```
$_long_variable_name; // valid
```

```
$2345; // invalid
```

Променливи

■ Променливи променливи

- Променлива, чието име се съдържа в друга променлива

```
$name = 'foo';  
$$name = 'bar'; // $foo има стойност 'bar'
```

- Тази техника позволява създаването на променливи, чиито имена не се подчиняват на изброените правила

```
$name = '123';  
$$name = '456';  
echo ${'123'}; // отпечатва '456'
```

- Изисква изключително внимание при употреба

Променливи

- Обхват на действие (variable scope)
 - Контекстът, в който променливата е дефинирана
 - Локален vs. глобален обхват

```
<?php
$a = 1; $b = 2; // глобален обхват

function foobar() {
    global $b;
    print $a; // локален обхват, $a има стойност NULL
    print $b; // глобален обхват на видимост
}
?>
```

Променливи

- Супер-глобални променливи (Superglobals)
 - Пре-дефинирани променливи, винаги достъпни - с глобална видимост, без да го указваме изрично:
 - `$_GET`
 - `$_POST`
 - `$_COOKIE`
 - `$_FILES`
 - `$_SERVER`
 - `$_ENV`
 - `$_REQUEST`
 - `$_SESSION`

Променливи

- Проверка за съществуването на дадена променлива
 - `isset($a)`
 - връща TRUE ако променливата съществува и има стойност различна от NULL
 - `is_null($a)`
 - връща TRUE ако променливата има стойност NULL

Променливи

Проверка за типа на дадена променлива

- `var_dump()`
 - за проверка на типа и стойността на даден израз
- `gettype()`
 - връща 'boolean', 'integer', 'double', 'string', 'array', 'object', 'resource', 'NULL', 'unknown type'
- `is_string()`
- `is_int()`
- `is_float()`
- `is_numeric()`
 - проверява, дали променливата е число или числов низ (напр. +0123.45e6)
- `is_object()`
- `is_array()`
- `is_resource`

Променливи

- Промяна на типа на променлива

```
settype($a, 'string')
```

- Преобразуване на типа на променлива (type casting)

```
$foo = 10;
```

```
$bar = (boolean) $foo // $bar holds TRUE
```

- (int), (integer), (bool), (boolean)
- (float), (double), (real)
- (string), (binary) // PHP 6+
- (array), (object)

- Автоматично преобразуване на типовете

```
$foo = "0"; // низ
```

```
$foo += 2; // цяло число (2)
```

```
$foo = $foo + 1.3 // реално число (3.3)
```

```
$foo = 5 + "10 piggies" // цяло число (15)
```

```
$bar = (boolean) $foo // $bar holds TRUE
```

Изрази (expressions)

- Всичко, което може да се изчисли като стойност
- PHP е изразно-ориентиран език - почти всяка конструкция е израз
- Елементарни изрази
 - Литерали - изчисляват се като собствената си стойност
 - Променливи - изчисляват се като стойността, запазена в променливата
- Съставни изрази
 - Формират се с помощта на елементарни изрази и оператори

Оператори

- Символ или поредица от символи, които в комбинация с една или повече стойности изпълняват някакво действие и обикновено като резултат се получава нова стойност

`$a = 1; // говорим за 3 стойности`

- Стойността на променливата `$a`, която ще се промени на "1"
- Стойността на целочислената константа "1"
- Стойността на самото присвояване като израз - изчислява се на присвоената стойност "1"

`$b = ($a = 1); $b е равно на "1"`

Оператори

- Според броя на операндите разглеждаме 3 вида оператори
 - Унарни - оперират върху един операнд

`!$foo; // Оператор за отрицание`

- Двоични - два операнда

`$a = 4; // Оператор за присвояване`

- Троен оператор

`($a) ? (1) : (2)`

`// Ако стойността на $a е TRUE, целия израз се изчислява`

`// като 1, в противен случай - 2`

Оператори

- Функционално разглеждаме следните видове
 - Оператори за присвояване
 - Аритметични оператори
 - Оператори за низове
 - Оператори за сравнение
 - Логически оператори
 - Побитови оператори
 - Оператори за контрол на грешките
 - Оператори за изпълнение
 - Инкрементиращи / декрементиращи оператори
 - Оператори за типове
 - Оператори за масиви

Оператори

■ Приоритет на операторите

- Редът, в който операторите в един израз се оценяват се определя от техния относителен приоритет

$3 + 2 * 5$

// операторът за умножение има по-висок приоритет

■ Асоциативност на операторите

- Редът, в който операторите с един и същи приоритет се оценяват се определя от тяхната асоциативност

$2 / 2 * 2 // 2$, понеже '*' и '/' са с лява асоциативност

$2 / (2 * 2) // 0.5$

$(2 / 2) * 2 // 2$

Оператори

Асоциативност	Оператори	Информация
без асоциативност	new	new
лява	[array()
без асоциативност	++ --	инкрементиране/декрементиране
без асоциативност	! ~ - (int) (float) (string) (array) (object) @	типове
без асоциативност	instanceof	типове
дясна	!	логически
лява	* / %	аритметични
лява	+ - .	аритметични и низови
лява	<< >>	побитови
без асоциативност	< <= > >=	сравнителни
без асоциативност	== != === !==	сравнителни
лява	&	побитови и референции
лява	^	побитови
лява		побитови
лява	&&	логически
лява		логически
лява	? :	третични
дясна	= += -= *= /= .= %= &= = ^= <<= >>=	присвоителни
лява	and	логически
лява	xor	логически
лява	or	логически
лява	,	множество употреби

`$a = 1; $b = 2;`

`$a = $b += 3 // дясна асоциативност: $a = 5; $b = 5`

Оператори

■ Оператори за присвояване

- Служат за присвояване на стойност към променлива

```
$variable = 'value';
```

- Най-простият присвоителен оператор е “=”

- Възможно е да се комбинира с почти всеки двоичен аритметичен или побитов оператор за да се извърши действие с променлива и едновременно с това да ѝ се присвои новата стойност

```
$variable = 1;
```

```
$variable += 3 // $variable вече има стойност 4
```

Оператори

- Аритметични оператори

Пример	Наименование	Резултат
$-\$a$	Отрицание	Противоположното на $\$a$.
$\$a + \b	Събиране	Сумата на $\$a$ и $\$b$.
$\$a - \b	Изваждане	Разликата между $\$a$ и $\$b$.
$\$a * \b	Умножение	Произведението на $\$a$ и $\$b$.
$\$a / \b	Деление	Частното на $\$a$ и $\$b$.
$\$a \% \b	Остатък от деление	Остатъкът от $\$a$ делено на $\$b$

Оператори

- Оператори за низове
 - Служат за съединяване на низове

```
$string = 'foo' . 'bar'; // 'foobar'  
$string2 = 'baz';
```

```
$string .= $string2; // 'foobarbaz'
```

Оператори

- Оператори за сравнение

Пример	Наименование	Резултат
<code>\$a == \$b</code>	Равно	TRUE ако \$a е равна на \$b.
<code>\$a === \$b</code>	Идентично	TRUE ако \$a е равна на \$b и ако са от един и също тип. (въведено в PHP 4)
<code>\$a != \$b</code>	Не-равно	TRUE ако \$a не е равна на \$b.
<code>\$a <> \$b</code>	Не-равно	TRUE ако \$a не е равна на \$b.
<code>\$a !== \$b</code>	Не-идентично	TRUE ако \$a не е равна на \$b или ако не са от един и същи тип. (въведено в PHP 4)
<code>\$a < \$b</code>	По-малко	TRUE ако \$a е строго по-малка от \$b.
<code>\$a > \$b</code>	По-голямо	TRUE ако \$a е строго по-голяма от \$b.
<code>\$a <= \$b</code>	По-малко или равно	TRUE ако \$a е по-малка или равна на \$b.
<code>\$a >= \$b</code>	По-голямо или равно	TRUE ако \$a е по-голяма или равна на \$b.

- Сравняване на стойности от други типове

`'ABC' > 'ABD' // FALSE`

`'apple' > 'Apple' // TRUE`

Оператори

- Логически оператори

Пример	Наименование	Резултат
<code>\$a and \$b</code>	И	TRUE ако и <code>\$a</code> и <code>\$b</code> са TRUE.
<code>\$a or \$b</code>	Или	TRUE ако <code>\$a</code> или <code>\$b</code> е TRUE.
<code>\$a xor \$b</code>	Изключващо или	TRUE ако или <code>\$a</code> или <code>\$b</code> е TRUE, но не и двете
<code>! \$a</code>	Не	TRUE ако <code>\$a</code> не е TRUE.
<code>\$a && \$b</code>	И	TRUE ако и <code>\$a</code> и <code>\$b</code> са TRUE.
<code>\$a \$b</code>	Или	TRUE ако <code>\$a</code> или <code>\$b</code> е TRUE.

- Комбинират 1 или 2 булеви стойности и като резултат от израза се получава 3-та, отново булева стойност
- Ако левият операнд на 'AND' се оцени като FALSE, интерпретаторът изобщо няма да оценява десния
- 1 унарн и 3 двоични оператора
- Причината за съществуването, както на 'and' и 'or', така и '&&' и '||' се крие в различния им приоритет. Следващите примери са еквивалентни:

`$x and $y || $z`

`$x && ($y || $z)`

`$x and ($y or $z)`

Оператори

- Побитови оператори

Пример	Наименование	Резултат
$\$a \& \b	И (And)	Вдигат се битовете, които са установени и в $\$a$ и в $\$b$.
$\$a \wedge \b	Изключващо или (Xor)	Вдигат се битовете, които са установени само в $\$a$ или само в $\$b$, но не и в двете едновременно.
$\sim \$a$	Не (Not)	Вдигат се битовете, които не са установени в $\$a$, и обратното - тези, които са установени, се свалят.
$\$a \ll \b	Преместване наляво (Shift left)	Преместване битовете на $\$a$ с $\$b$ позиции наляво (всяка позиция означава "умножение по две")
$\$a \gg \b	Преместване надясно (Shift right)	Преместване битовете на $\$a$ с $\$b$ позиции наляво (всяка позиция означава "деление на две")

- Работят с двоичните репрезентации само на целочислени стойности. Ако някой от операндите не е целочислен, той ще бъде конвертиран до цяло число преди действието на побитовия оператор

```

1 & 9 // 1
1001 (9)
0001 (1)
----
0001 (1)

```

Оператори

- Оператор за подтискане на грешките
 - Добавен пред израз подтиска извеждането на почти всички грешки от PHP интерпретатора

```
$x = @mysql_connect();
```
 - Някои библиотеки не използват интерпретатора за да показват грешки, а го заобикалят, като сами ги извеждат. Такива грешки няма да бъдат подтиснати чрез този оператор.
- Оператор за изпълнение на конзолна команда

```
$output = `ls -la`;
```

 - Аналогичен с извикване на функцията `shell_exec()`

Оператори

- Инкрементиращи / декрементиращи оператори

Пример	Наименование	Резултат
++\$a	Предварително инкрементиране	Увеличава \$a с едно, след което връща \$a.
\$a++	Последващо инкрементиране	Връща \$a, след което увеличава \$a с едно.
--\$a	Предварително декрементиране	Намалява \$a с едно, след което връща \$a.
\$a--	Последващо декрементиране	Връща \$a, след което намалява \$a с едно.

- Приложими са само върху променливи
- Инкрементиране на нулеви стойности връща 1, декрементирането им не е възможно
- Низови стойности (само от ASCII диапазона) се инкрементират, но не се декрементират

```
$str = 'Y';
echo ++$str; // 'Z'
echo ++$str; // 'AA'
echo ++$str; // 'AB'
```

Оператори

- Оператори за масиви

Пример	Наименование	Резултат
<code>\$a + \$b</code>	Обединение	Обединението на \$a и \$b.
<code>\$a == \$b</code>	Равенство	TRUE ако \$a и \$b имат едни и същи двойки ключ/стойност.
<code>\$a === \$b</code>	Идентичност	TRUE ако \$a и \$b имат едни и същи двойки ключ/стойност в един и същи ред и са от един и същи тип.
<code>\$a != \$b</code>	Неравенство	TRUE ако \$a не е равно на \$b.
<code>\$a <> \$b</code>	Неравенство	TRUE ако \$a не е равно на \$b.
<code>\$a !== \$b</code>	Неидентичност	TRUE ако \$a не е идентично на \$b.

```
$a = array('0' => 'zero', '1' => 'one');
$b = array('0' => 'foo', '1' => 'bar', '2' => 'foobar');
var_dump($a + $b);
```

```
array(3) {
    [0]=>
    string(4) "zero"
    [1]=>
    string(3) "one"
    [2]=>
    string(6) "foobar"
}
```

Въпроси?

**Благодаря ви
за вниманието!**