LEKSYKON KIESZONKOWY

PHP 5

Praktyczna pomoc na co dzień — sięgnij po świetną ściągawkę z PHP!

Struktura języka, czyli znaczniki, typy danych, operatory i inne elementy

Instrukcje sterujące oraz funkcje, czyli wydawanie różnych poleceń

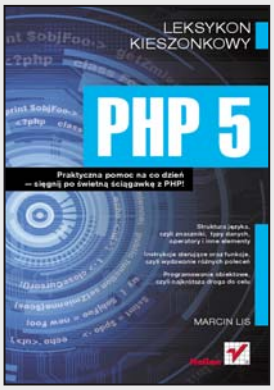
Programowanie obiektowe, czyli najkrótsza droga do celu

MARCIN LIS

Helion

PHP 5. Leksykon kieszonkowy

Autor: Marcin Lis
ISBN: 978-83-246-3394-4
Format: 122×194, stron: 168



Praktyczna pomoc na co dzień – sięgnij po świetną ściągawkę z PHP!

- Struktura języka, czyli znaczniki, typy danych, operatory i inne elementy
- Instrukcje sterujące oraz funkcje, czyli wydawanie różnych poleceń
- Programowanie obiektowe, czyli najkrótsza droga do celu

Język skryptowy PHP od lat wciąż na nowo udowadnia swoją elastyczność i niezawodność. Wykorzystywany do tworzenia dynamicznych serwisów internetowych, a także samodzielnych aplikacji, szczególnie dobrze sprawdza się wtedy, gdy chcemy zapewnić możliwość współdziałania naszej witryny z różnymi formatami danych oraz skomplikowanymi bazami. Ponadto jego opanowanie nie nastręcza wielu trudności, a efekty pracy nawet początkujących webmasterów bywają wręcz spektakularne. Jeśli masz już za sobą pierwsze kroki w dziedzinie programowania w PHP5, w codziennej pracy z pewnością przyda Ci się poręczna ściąga, w której zawarto najważniejsze informacje i sposoby wykonywania podstawowych zadań programistycznych. Właśnie trzymasz ją w ręku!

„PHP5. Leksykon kieszonkowy" powstał po to, by ułatwić Ci szybkie przypomnienie sobie wybranych zagadnień związanych z tą tematyką. Znajdziesz tu informacje o instalacji oraz konfiguracji środowiska w systemach Linux i Windows, strukturze języka, instrukcjach sterujących czy funkcjach. Przeczytasz o operacjach na danych, a także o obsłudze ciągu znaków daty i czasu. Przypomnisz sobie, jak używać tablic, współpracować z serwerem i przeglądarką, obsługiwać wyjątki. W leksykonie odszukasz także wiadomości dotyczące systemu plików i programowania obiektowego. Wszystko to wydatnie skróci Twoją pracę i pomoże Ci uniknąć wielu potencjalnych błędów.

- Instalacja i konfiguracja
- Struktura języka
- Instrukcje sterujące
- Funkcje
- Operacje na danych i obsługa ciągu znaków
- Tablice
- Obsługa daty i czasu
- Współpraca z serwerem i przeglądarką
- Programowanie obiektowe
- Wyjątki
- Obsługa systemu plików

Dobra podpowiedź zawsze pod ręką!

Helion SA
ul. Kościuszki 1c
44-100 Gliwice
tel. 32 230 98 63
e-mail: helion@helion.pl

Spis treści

Rozdział 4. Funkcje

Deklaracje

W celu utworzenia funkcji należy użyć słowa kluczowego `function`. Taka konstrukcja ma schematyczną postać:

```
function nazwa_funkcji()
{
  //instrukcje wnętrza funkcji
}
```

Przy nazywaniu funkcji obowiązują takie same zasady jak przy innych identyfikatorach. Jeżeli funkcja ma przyjmować argumenty, ich listę należy umieścić w nawiasie okrągłym występującym za nazwą:

```
function nazwa_funkcji($argument1, $argument2,…,$argumentN)
{
  //instrukcje wnętrza funkcji
}
```

W ciele (we wnętrzu) funkcji można zawrzeć dowolny, poprawny składniowo, kod PHP, włącznie z definicjami innych funkcji i klas.

Oto przykład funkcji wyświetlającej napis:

```
function wyświetl()
{
  echo "Tekst generowany przez funkcję wyświetl.";
}
```

Zwracanie wartości

Zwracanie wartości przez funkcję jest realizowane za pomocą instrukcji `return`. Jeśli wystąpi ona wewnątrz funkcji, ta jest przerywana i zwraca wartość występującą po `return`. Schematycznie tego typu konstrukcja wygląda następująco:

```
function nazwa_funkcji(argumenty)
{
  //instrukcje wnętrza funkcji
  return wartość;
}
```

W miejscu wywołania takiej funkcji zostanie wstawiona zwrócona przez nią wartość, która będzie mogła być wykorzystana w dalszej części

skryptu. Jeżeli zostanie użyta instrukcja `return` bez wskazanej wartości, funkcja jest przerywana i zwracana wartość `null`. Jeśli funkcja nie zawiera instrukcji `return`, zwracaną wartością jest również `null`.

Oto przykład funkcji zwracającej wartość arytmetyczną całkowitoliczbową (`integer`) `12`:

```
function func()
{
  return 12;
}
```

A to przykład funkcji przyjmującej dwa argumenty i zwracającej wynik ich dodawania:

```
function dodaj($arg1, $arg2)
{
  return $arg1 + $arg2;
}
```

Wywoływanie funkcji

Aby wywołać funkcję, należy podać jej nazwę zakończoną nawiasem okrągłym. Jeżeli funkcja przyjmuje argumenty, należy je umieścić w tym nawiasie. W miejscu wywołania funkcji podstawiana jest zwrócona przez nią wartość. Przykładowa instrukcja:

```
$wynik = func();
```

spowoduje wywołanie funkcji o nazwie `func` i przypisanie wyniku jej działania zmiennej `$wynik`. Oto przykład wywołania funkcji z dwoma argumentami:

```
$wynik = dodaj(12, 24);
```

Sposoby przekazywania argumentów

Funkcja może mieć dowolną liczbę argumentów oddzielonych od siebie znakiem przecinka. Argumenty przekazywane są na dwa sposoby:

- przez wartość (z ang. *by value*),
- przez referencję (z ang. *by reference*),

Argumenty mogą mieć też wartości domyślne. Standardowo przekazywanie odbywa się przez wartość, podobnie jak w innych językach programowania.

Przekazywanie przez wartość

Przekazywanie przez wartość oznacza, że funkcja otrzymuje kopie argumentów źródłowych i wszelkie operacje wykonuje na tych kopiach. Nie jest zatem w stanie dokonać żadnej modyfikacji oryginału. Jeżeli więc jako argument została przekazana zmienna, to jej stan nie zmieni się, niezależnie od tego, co zostanie wykonane wewnątrz funkcji:

```
<?php
function f($arg){
  //modyfikacja jedynie kopii danych
  //nie ma wpływu na wartość zmiennej $zmienna
  $arg = 'dwa';
}
$zmienna = 'jeden';
echo $zmienna, "\n";
f($zmienna);
//zmienna ma wartość 'jeden'
echo $zmienna;
?>
```

Przekazywanie przez referencję

Przekazywanie przez referencję oznacza przekazanie do funkcji oryginalnych danych. Jeżeli funkcja zmieni ich wartość (wartości), pozostanie ona zmieniona również po zakończeniu działania funkcji. Aby skorzystać z tego sposobu przekazywania argumentów, dany argument należy poprzedzić znakiem & (ampersand), schematycznie:

```
function funkcja(&argument);
```

np.:

```
<?php
function f(&$arg){
  //tutaj następuje modyfikacja
  //wartości zmiennej $zmienna
  $arg = 'dwa';
}
$zmienna = 'jeden';
echo $zmienna, "\n";
f($zmienna);
//zmienna ma wartość 'dwa'
echo $zmienna;
?>
```

Domyślne wartości argumentów

Argumenty domyślne są definiowane podobnie jak w innych językach programowania. Należy skorzystać z konstrukcji o schematycznej postaci:

```
function nazwa_funkcji ($argument1 = wartość, $argument2 =
↪wartość2, ..., $argumentN = wartośćN)
{
  //treść funkcji
}
```

Wartość domyślna musi być wyrażeniem stałym (o stałej wartości, z ang. *constant expression*), nie może więc to być np. zmienna. Należy również pamiętać, że konieczne jest podanie wszystkich domyślnych argumentów z prawej strony listy argumentów przed wszystkimi argumentami zwykłymi, np.:

```
function show($arg1, $arg2 = "abc", $arg3 = "def")
{
  echo $arg1. " ".$arg2." ".$arg3;
}
```

Po takiej deklaracji funkcja `show` mogłaby być wywołana na jeden z trzech poniższych sposób:

```
show("123");
show("123", "456");
show("123", "456", "789");
```

Zmienna liczba argumentów

W funkcjach można stosować zmienną liczbę argumentów. Nie ma do tego żadnych specjalnych konstrukcji. Funkcja może być wywołana z dowolną liczbą argumentów, większą niż liczba podana w deklaracji. Do obsługi zmiennej listy argumentów wykorzystywane są trzy funkcje:

- `func_num_args` — zwraca liczbę argumentów funkcji,
- `func_get_arg` — zwraca argument o podanym numerze,
- `func_get_args` — zwraca listę argumentów w postaci tablicy.

Oto przykład skryptu zawierającego funkcję o zadeklarowanej zerowej liczbie argumentów, która dokonuje łączenia łańcuchów znakowych przekazanych jej w postaci argumentów:

```
<?php
function połącz(){
  $val = "";
  $count = func_num_args();
  for($i = 0; $i < $count; $i++){
    $val .= func_get_arg($i);
  }
  return $val;
}
$str = połącz("To ", "jest ", "test");
echo $str;
?>
```

A to ten sam skrypt wykorzystujący funkcję `func_get_args` i pętlę `foreach`:

```
<?php
function połącz(){
  $val = "";
  $tab = func_get_args();
  foreach($tab as $v){
    $val .= $v;
  }
  return $val;
}
$str = połącz("To ", "jest ", "test");
echo $str;
?>
```

Zasięg zmiennych

W PHP zasięg zmiennej jest ograniczony do kontekstu, w którym została zdefiniowana. W tym znaczeniu zmienne można podzielić na:

- globalne,
- lokalne.

Zmienne globalne

Zmienna zadeklarowana w skrypcie poza ciałem funkcji lub klasy jest zmienną globalną, tzn. dostępną bezpośrednio w każdym miejscu skryptu poza wnętrzami funkcji. W przykładowym kodzie:

```
<?php
$liczba = 100;
function f()
{
  echo $liczba;
}
f();
?>
```

zmienna `$liczba` jest globalna i nie ma do niej dostępu w funkcji `f`. Dlatego też po wywołaniu funkcji wartość nie zostanie wyświetlona, wygenerowane zostanie natomiast ostrzeżenie (poziom `notice`) o niezdefiniowanej zmiennej `$liczba`.

Aby w funkcji uzyskać dostęp do zmiennych o zasięgu globalnym, należy użyć słowa kluczowego `global` lub tablicy `$GLOBALS`. W pierwszym przypadku przed odwołaniem (najlepiej na początku kodu funkcji) należy użyć konstrukcji o schematycznej postaci:

```
global $zmienna1, $zmienna2,…,$zmiennaN;
```

Po wykonaniu takiej instrukcji będzie można się odwoływać do wszystkich zmiennych globalnych wymienionych po `global`, np.:

```
<?php
$liczba = 100;
function f()
{
  global $liczba;
  echo $liczba;
}
f();
?>
```

W drugim przypadku, ponieważ tablica `$GLOBALS` zawiera odwołania do wszystkich zmiennych globalnych skryptu, należy skorzystać z odwołania typu:

```
$GLOBALS['nazwa_zmiennej']
```

np.:

```
<?php
$liczba = 100;
function f()
{
  echo $GLOBALS['liczba'];
}
f();
?>
```

Zmienne lokalne

Zasięg zmiennych lokalnych jest ograniczony wyłącznie do wnętrza funkcji, w której zostały zdefiniowane. Odwołania w innym miejscu skryptu nie są możliwe, np.:

```
<?php
function f()
{
  $liczba = 100;
  echo "Wewnątrz funkcji f: $liczba\n";
}
f();
echo "Poza funkcją f: $liczba";
?>
```

Zmienne statyczne

Zmienne statyczne to zmienne lokalne funkcji, które zachowują swoją wartość pomiędzy jej wywołaniami. Aby zadeklarować taką zmienną, należy użyć słowa `static`, schematycznie:

```
static $nazwa_zmiennej = wartość;
```

Po takiej deklaracji (we wnętrzu funkcji) pierwsze wywołanie funkcji spowoduje utworzenie zmiennej statycznej i zapamiętanie jej ostatniej wartości, natomiast w każdym kolejnym wywołaniu instrukcja przypisująca pierwotną wartość zmiennej będzie ignorowana, a użyta zostanie wartość z poprzedniego wywołania, np.:

```
<?php
function f($val)
{
  static $liczba = 100;
  $liczba += $val;
  echo "Wartość zmiennej : $liczba\n";
}
f(10);
f(20);
?>
```

Wartość zmiennej jako nazwa funkcji

W PHP wartość zmiennej może być potraktowana jako nazwa funkcji do wywołania. Aby skorzystać z takiej techniki, należy za nazwą zmiennej umieścić nawias okrągły, schematycznie:

```
$nazwa_zmiennej();
```

Jeżeli zatem zmiennej `$nazwa` zostanie przypisany ciąg znaków zawierający nazwę istniejącej funkcji `wyświetl`, to po użyciu instrukcji:

```
$nazwa();
```

funkcja `wyświetl` zostanie wywołana:

```
<?php
function wyświetl()
{
  echo "To jest funkcja wyświetl.";
}

$nazwa = 'wyświetl';

$nazwa();

?>
```

W ten sam sposób mogą być wywoływane metody obiektów. Jeżeli funkcja lub metoda wymaga podania argumentów, należy je podać w nawiasie okrągłym występującym za nazwą zmiennej, np.:

```
<?php
function dodaj($arg1, $arg2)
{
  return $arg1 + $arg2;
}
```

```
$nazwa = 'dodaj';

$wynik = $nazwa(2, 3);
echo $wynik;
?>
```

Definiowanie funkcji wewnątrz funkcji

W PHP można definiować funkcje wewnątrz innych funkcji. Powstają wtedy funkcje wewnętrzne dostępne tylko w obrębie (w zasięgu) funkcji zewnętrznej (głównej). Nie ma limitu poziomu zagnieżdżenia funkcji wewnętrznych (tzn. jedna funkcja wewnętrzna może zawierać kolejną funkcję wewnętrzną). Schematycznie konstrukcja funkcji wewnętrznej wygląda następująco:

```
function nazwa_funkcji_zewnętrznej(argumenty){
  function nazwa_funkcji_wewnętrznej(argumenty){
    //treść funkcji wewnętrznej
  }
  //dalsza treść funkcji zewnętrznej
}
```

Oto przykład użycia funkcji wewnętrznych:

```
<?php
function działanie($val1, $val2, $op)
{
  function dodaj($arg1, $arg2){
    return $arg1 + $arg2;
  }
  function odejmij($arg1, $arg2){
    return $arg1 - $arg2;
  }
  switch($op){
    case '+': return dodaj($val1, $val2);
    case '-': return odejmij($val1, $val2);
    default: return null;
  }
}

$wynik = działanie(2, 3, '+');
echo $wynik;
?>
```

Funkcje anonimowe

Od PHP 5.3.0 dostępne są funkcje anonimowe, czyli takie, które nie posiadają nazwy. Najczęściej używane są podczas stosowania funkcji zwrotnych (z ang. *callback functions*). Definicja funkcji anonimowej wygląda podobnie jak definicja funkcji zwykłej, schematycznie:

```
function (argument1, argument2,…, argumentN)
{
  //treść funkcji
}
```

Oto przykład użycia funkcji anonimowej jako funkcji zwrotnej:

```
<?php
function działanie($val1, $val2, $func)
{
  return $func($val1, $val2);
}

$wynik = działanie(2, 3,
 function($arg1, $arg2){
   return $arg1 + $arg2;
 }
);

echo $wynik;
?>
```

W tym przypadku trzecim argumentem funkcji `działanie` jest funkcja anonimowa przyjmująca dwa argumenty (`$arg1` i `$arg2`) i zwracająca wynik ich dodawania. Funkcja anonimowa jest wywoływana w funkcji `działanie` przez zastosowanie składni opisanej w podrozdziale „Wartość zmiennej jako nazwa funkcji”.

Funkcja anonimowa może być również przypisana bezpośrednio zmiennej, np.:

```
<?php
$zmienna = function($arg1){
  return $arg1 * 2;
};
$wynik = $zmienna(12);
echo $wynik;
?>
```

Skorowidz

A

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

LEKSYKON KIESZONKOWY

PHP 5

- Instalacja i konfiguracja
- Struktura języka
- Instrukcje sterujące
- Funkcje
- Operacje na danych i obsługa ciągu znaków
- Tablice
- Obsługa daty i czasu
- Współpraca z serwerem i przeglądarką
- Programowanie obiektowe
- Wyjątki
- Obsługa systemu plików

Język skryptowy PHP od lat wciąż na nowo udowadnia swoją elastyczność i niezawodność. Wykorzystywany do tworzenia dynamicznych serwisów internetowych, a także samodzielnych aplikacji, szczególnie dobrze sprawdza się wtedy, gdy chcemy zapewnić możliwość współdziałania naszej witryny z różnymi formatami danych oraz skomplikowanymi bazami. Ponadto jego opanowanie nie nastręcza wielu trudności, a efekty pracy nawet początkujących webmasterów bywają wręcz spektakularne. Jeśli masz już za sobą pierwsze kroki w dziedzinie programowania w PHP 5, w codziennej pracy z pewnością przyda Ci się poręczna ściąga, w której zawarto najważniejsze informacje i sposoby wykonywania podstawowych zadań programistycznych. Właśnie trzymasz ją w ręku!

„PHP 5. Leksykon kieszonkowy" powstał po to, by ułatwić Ci szybkie przypomnienie sobie wybranych zagadnień związanych z tą tematyką. Znajdziesz tu informacje o instalacji oraz konfiguracji środowiska w systemach Linux i Windows, strukturze języka, instrukcjach sterujących czy funkcjach. Przeczytasz o operacjach na danych, a także o obsłudze ciągu znaków daty i czasu. Przypomnisz sobie, jak używać tablic, współpracować z serwerem i przeglądarką, obsługiwać wyjątki. W leksykonie odszukasz także wiadomości dotyczące systemu plików i programowania obiektowego. Wszystko to wydatnie skróci Twoją pracę i pomoże Ci uniknąć wielu potencjalnych błędów.

Dobra podpowiedź zawsze pod ręką!

helion.pl
księgarnia internetowa

Nr katalogowy: 6793

Księgarnia internetowa:
http://helion.pl

Zamówienia telefoniczne:
0 801 339900
0 601 339900

Helion
Sprawdź najnowsze promocje:
http://helion.pl/promocje
Książki najchętniej czytane:
http://helion.pl/bestsellery
Zamów informacje o nowościach:
http://helion.pl/nowosci

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

Cena 24,90 zł
ISBN 978-83-246-3394-4
9 788324 633944

Informatyka w najlepszym wydaniu