Funkcje

Czym jest funkcja?

Funkcje w PHP są częściowo podobne do funkcji matematycznych. Mogą przyjmować argumenty oraz zwracać wartości. Nie jest to jednak wymogiem. Funkcją nazywamy napisany przez nas kod, zamknięty w nawiasy klamrowe, poprzedzony słowem kluczowym ***function*** oraz unikatową nazwą.

Funkcja bezargumentowa, nie zwracająca wartości.

Zobaczmy poniższy listing:

```
<?php

function wyswietl_powitanie() // deklaracja funkcji
{
   echo "Witam serdecznie!";  // ciało funkcji, czyli
   echo "Proszę się zarejestrować.";  // instrukcje do wykonania
}

?>
```

Przeanalizujmy napisany kod. Przedstawia funkcję, której zadaniem jest wyświetlenie na ekranie dwóch komunikatów. Jest to jedynie deklaracja. Żeby użyć tak napisanej funkcji należy ją wywołać. Wywołanie przedstawione jest poniżej:

```
<?php

// pozostały kod w pliku
wyswietl_powitanie();
// pozostały kod w pliku
wyswietl_powitanie();
wyswietl_powitanie();
// pozostały kod

?>
```

Jak widać, żeby użyć napisanej funkcji, wystarczy napisać jej nazwę. Jak każde polecenie w PHP, tak również wywołanie napisanej przez nas funkcji, musi być zakończone średnikiem. Co znaczą nawiasy między nazwą, a średnikiem? Jest to miejsce na podanie argumentów funkcji. Skoro nasza funkcja nie przyjmuje żadnych, są one puste. Nie należy jednak zapominać o ich umieszczeniu, nawet gdy funkcja jest bezargumentowa!

Funkcja zwracająca wartość

Poprzednio omawialiśmy funkcje, których wykonywane zadanie zamykało się wewnątrz struktury. Znaczy to tyle, że nie miały wpływu na wykonanie dalszego kodu. Tym razem zajmiemy się funkcjami, które coś wnoszą do programu. Żeby funkcja zwróciła wartość do programu głównego, musimy umieścić zwracane wyrażenie po słowie return.

Zobaczmy przykład:

```
<?php

function tresc_powitania() // deklaracja funkcji
{
   return "Witam wszystkich!";
}

$powitanie = tresc_powitania();
echo $powitanie;

?>
```

Żeby zrozumieć zasadę działania funkcji zwracającej wartość, wyobraźmy sobie, że funkcja to taka zmienna, której wartość zmienia się dynamicznie. Co za tym idzie, funkcję możemy przypisać zmiennej. Dodatkowo możemy sprawdzić, czy funkcja jest mniejsza lub większa od pewnej liczby. Oczywiście, mówiąc funkcja, mamy na myśli wartość zwracaną przez daną funkcję.

Przeanalizujmy skrypt:

```
<?php

function oblicz()
{
   $zm1 = 3;
   $zm1 += 5;
   $zm1++;
   return $zm1;
}

if (oblicz() > 5)
   echo "Funkcja zwraca wartość większą od 5";
else
   echo "Wartość zwracana przez funkcję jest mniejsza od 6";

?>
```

Funkcja przyjmująca argumenty

Co nazywamy argumentami? Wszystkie wartości przekazywane w nawiasie, zaraz po nazwie funkcji. Jest to bardzo przydatna rzecz, która w dużej mierze usprawnia programowanie. Zmienna przekazana jako argument może być używana i modyfikowana wewnątrz funkcji, bez wpływu na jej wartość w programie głównym.

Przykład zastosowania:

```
<?php

function przywitaj($zmienna_z_imieniem)
{
   echo 'Witaj '.$zmienna_z_imieniem.'!';
}

$imie = "Marcin";

przywitaj($imie);

?>
```

Kilka słów wyjaśnień. Tworząc deklarację funkcji, podajemy jako argumenty fikcyjne nazwy zmiennych. Następnie, wywołując funkcję, podajemy istniejącą zmienną, która jest podstawiana pod zmienną fikcyjną. Podsumowując, działa to tak, że wszędzie, gdzie użyta była $zmienna_z_imieniem, użyta zostanie $imie. Funkcja wyświetli "Witaj Marcin!".

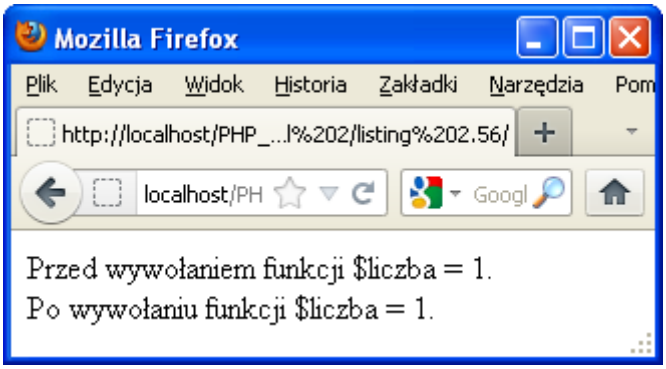
Sposoby przekazywania argumentów

Argumenty funkcji mogą być przekazywane na dwa sposoby:

- Przez wartość
- Za pomocą referencji

W prezentowanych do tej pory przykładach wykorzystywane było domyślne przekazywanie argumentów przez wartość. Oznacza to, że w rzeczywistości funkcja otrzymuje kopię argumentów źródłowych i wszelakie operacje wykonuje na tych kopiach. Nie jest więc w stanie dokonać żadnej modyfikacji oryginału.

```
1 <?php
2 function dodajJeden($liczba)
3 {
4   $liczba = $liczba + 1;
5 }
6 $liczba = 1;
7 echo "Przed wywołaniem funkcji \$liczba = $liczba. <br />";
8
9 dodajJeden($liczba);
10
11 echo "Po wywołaniu funkcji \$liczba = $liczba. <br />";
12
13 ?>
14
```



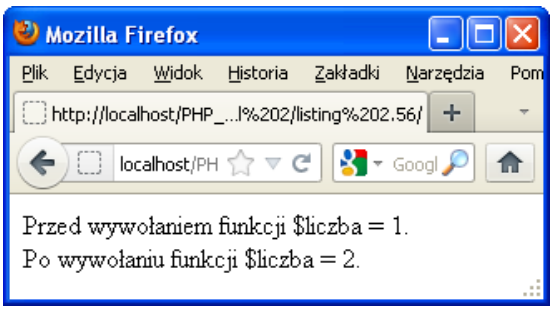
Jeśli jednak chcemy mieć możliwość modyfikowania w funkcji argumentu oryginalnego, trzeba zastosować sposób drugi – przekazywanie przez referencję. W tym celu przed nazwą argumentu należy umieścić znak & (ampersand).

```php
<?php
function dodajJeden(&$liczba)
{
  $liczba = $liczba + 1;
}
$liczba = 1;
echo "Przed wywołaniem funkcji \$liczba = $liczba. <br />";

dodajJeden($liczba);

echo "Po wywołaniu funkcji \$liczba = $liczba. <br />";

?>
```



Domyślne argumenty funkcji

PHP pozwala na konstruowanie funkcji o domyślnych wartościach argumentów. Dzięki temu część parametrów lub nawet wszystkie będą mogły być pominięte w wywołaniu, a w ich miejsce zostaną wstawione wartości zdefiniowane podczas tworzenia funkcji

```php
<?php
function dodaj($wartosc, $ile = 1)
{
  return $wartosc + $ile;
}
$liczba1 = 10;

$liczba2 = dodaj($liczba1, 5);
echo $liczba2 , '<br />';

$liczba2 = dodaj($liczba1);
echo $liczba2 , '<br />';

$liczba2 = dodaj(5, 3);
echo $liczba2, '<br />';

$liczba2 = dodaj(5);
echo $liczba2, '<br />';

?>
```

Funkcje rekurencyjne

Funkcją rekurencyjną nazywamy funkcję odwołującą się do siebie samej. Przykład powinien nieco rozjaśnić wątpliwości.

Zobaczmy listing:

```
<?php

function silnia($liczba)
{
   if($liczba < 2)
      return 1;
   else
      return $liczba*silnia($liczba-1);
}

echo silnia(5);

?>
```

Przeanalizujmy działanie kodu. Jeżeli liczba jest mniejsza od 2, czyli 0 lub 1, zwrócona zostanie wartość 1 (z definicji funkcji). Jeśli natomiast liczba jest większa, wywołujemy funkcję ponownie z argumentem pomniejszonym o 1. Wynika to z faktu, że 4! = 4 * 3!. Robimy tak dopóki nie zejdziemy do jedynki.

Plusy i minusy stosowania funkcji rekurencyjnych

Podstawową zaletą funkcji rekurencyjnych jest prostota kodu. Na funkcji obliczającej silnię nie widać tego tak znacząco, lecz pisząc bardziej rozbudowane konstrukcje, jest to zauważalne. Programiści jednak odchodzą od stosowania funkcji rekurencyjnych z racji dużej ilości pamięci, zajmowanej podczas kolejnych wywołań. Można sobie to łatwo zobrazować.

Licząc silnię z dziesięciu, musimy wywołać funkcję dziesięć razy (z argumentem: 10, 9, 8 itd.). Dodatkowo system musi zapamiętać wynik zwracany przez każdą z funkcji. Z tego powodu bardzo zachęcam do stosowania klasycznych funkcji. Na pewno usprawni to działanie strony

Zadanie

Napisz funkcje jednoargumentową, gdzie jako argument będzie pobierała ciąg znaków PESEL i z tego ciągu rozkoduje informacje zawartą w nim (data urodzenia i płeć) i wykorzystaj tą funkcje w przykładowym programie.