Tworzenie własnych funkcji

PHP 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.

Intencją tworzenia funkcji jest ich ponowne użycie. Żeby nie kopiować i nie wklejać tego samego kodu w kilku miejscach w pliku, piszemy go raz w postaci funkcji. Wtedy możemy go wykorzystywać podając jedynie jej nazwę. Tak samo łatwiej nam poprawić błąd w jednym miejscu (jeśli znajdziemy jakiś wewnątrz funkcji) niż poprawiać go w każdym miejscu skopiowanego skryptu.

Nazwy funkcji nie mogą się powtarzać. Nie można również używać nazw istniejących już poleceń, takich jak while, if, echo itp.

Funkcje możemy podzielić na cztery kategorie:

- bezargumentowa, niezwracająca wartości,
- bezargumentowa, zwracająca pewną wartość,
- przyjmująca argumenty, niezwracająca wartości,
- przyjmująca argumenty oraz zwracająca wartość.

Funkcja bezargumentowa, niezwracająca wartości
Zobaczm 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!”.

Funkcje kompleksowo

Przedstawię tutaj funkcje zarówno przyjmujące argumenty, jak i zwracające wartość. Takie zastosowanie funkcji jest powszechne i bardzo praktyczne. Zazwyczaj jest tak, że na podstawie danych wartości, szukamy wyniku.
Zobaczmy poniższy przykład:

```
<?php

function kwadrat($liczba)
{
  return $liczba*$liczba;
}

$numer = 5;
$wynik = kwadrat($numer);

echo $wynik; // wyświetli 25

?>
```

Widać jasno, że przyjmowanym argumentem jest liczba, a zwracana wartością jest jej kwadrat. Jest to bardzo proste przedstawienie działania funkcji. Na podstawie otrzymanych danych obliczany jest wynik.
Jeszcze jeden przykład:

```
<?php

function silnia($liczba)
{
  $wynik = 1;
  while($liczba > 1)
  {
    $wynik *= $liczba;
    $liczba--;
  }
  return $wynik;
}

?>
```

Powyższa funkcja oblicza silnie podanego argumentu. Silnię zapisujemy tak: 4! = 1*2*3*4, 0! = 1. Należy pamiętać, że silnia jest definiowana tylko dla liczb naturalnych. W przykładzie zakładamy, że podawany jest poprawny argument. Możesz spróbować lekko ją zmodyfikować, dodając warunek sprawdzający poprawność danych.

Funkcje rekurencyjne

Ostatnim razem napisaliśmy funkcję obliczającą silnię. Teraz zobaczymy, jak wykonać to zadanie za pomocą funkcji rekurencyjnej. 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

PHP Tablice

Przyszła pora na absolutny niezbędnik. Nie tylko w php, chyba w każdym języku programowania. Mowa tutaj o tablicach. Nie sposób wyobrazić sobie bez nich pracy. W skrócie, tablicą w PHP nazywamy kontener na wartości, do których odwołujemy się za pomocą indeksu. Jak wygląda to w praktyce?
Zobacz przykład poniżej:

```
<?php

$tablica[0] = 1;    // przypisanie
$tablica[1] = 4;    // wartości
$tablica[2] = 1;    // kolejnym
$tablica[3] = 0;    // indeksom

for ($i = 0; $i < 4; $i++) // wyświetlenie każdego
  echo $tablica[$i];  // indeksu za pomocą pętli for
?>
```

Kod wyświetli napis „1410”. Ten przykład pokazuje jedynie, w jaki sposób tworzy się tablicę i dodaje kolejne wartości. Tablic używamy głównie w dwóch przypadkach. Kiedy nie wiemy, ile wartości będziemy chcieli zapisać (czyli, ile zmiennych pojedynczych mamy utworzyć) oraz gdy ich liczba byłaby dość duża (wygodniej operować na jednej zmiennej tablicowej odwołując się indeksem, niż tworzyć np. 10 zmiennych jednoelementowych).
Praktyczny przykład. Chcemy znaleźć i zapisać wszystkie liczby podzielne przez 4 z przedziału od 0 do pewnej zmiennej. Do przechowania liczb użyjemy tablicy.
Przegląd rozwiązania:

```
<?php

$pewna_zmienna = X; // gdzie X to dowolna wartość

$j = 0; // zmienna pomocnicza
$i = 0; // zmienna iteracyjna

while($i <= $pewna_zmienna) // warunek kontynuacji pętli
{
  if($i % 4 == 0) // jeśli podzielna przez 4
  {
    $tablica[$j] = $i;  // dodaj kolejny element do tablicy
    $j++; // zwiększ indeks o 1
```

```
  }
  $i++; // zwiększamy $i o 1, aż przekroczymy $pewna_zmienna
}

for ($i = 0; $i < $j; $i++) // wyświetlenie wszystkich
  echo $tablica[$i]."<br/>"; // elementów tablicy

?>
```

Zasada działania powyższego programu jest bardzo prosta. Najpierw przypisujemy wartość **$pewnej_zmiennej**. Następnie tworzymy dwie zmienne pomocnicze – jedną odpowiedzialną za ilość przebiegów (**$i**) oraz drugą za indeksowanie tablicy. W momencie, gdy w pętli spełniony będzie warunek podzielności liczby przez 4 (zwraca resztę 0) do tablicy o indeksie **$j** przypisywana jest ta wartość, po czym zwiększamy numer indeksu o 1.
Wiesz już po co stosuje się tablice PHP i w jaki sposób ich używać. Jest to wiedza kluczowa, bardzo usprawniająca programowanie w PHP.

Tablice wielowymiarowe

Poprzednim tematem były tablice – bardzo praktyczna rzecz w programowaniu. Przechowywały one wartości pod przypisanymi indeksami. Czy istnieje możliwość przechowania całej tablicy jako jednej pozycji? Oczywiście, że tak! Mówimy wtedy o tablicy dwuwymiarowej. Można też stworzyć tablicę z tablic dwuwymiarowych – wtedy tablica ma trzy wymiary. Powszechnie jednak nie stosuje się tablic większych niż dwa wymiary.
W jakich przypadkach stosujemy dwuwymiarowe tablice? Gdy chcemy w uporządkowany sposób przechowywać zestawy danych. Wyobraźmy sobie, że mamy klasę uczniów. Każdy uczeń ma imię, nazwisko, adres oraz datę urodzenia. Dane każdego z uczniów trzymamy w tablicy. Następnie każdą z tych tablic przypisujemy do indeksu tablicy głównej. Powstaje nam wtedy „tablica tablic”.
Zobaczmy przykład:

```
<?php

$tablica_ucznia[0] = "Janek";
$tablica_ucznia[1] = "Kowalski";
$tablica_ucznia[2] = "14-10-1995";

$tablica_klasy[0] = $tablica_ucznia;

$tablica_ucznia[0] = "Krzysiek";
$tablica_ucznia[1] = "Nowak";
$tablica_ucznia[2] = "24-12-1994";

$tablica_klasy[1] = $tablica_ucznia;

$tablica_ucznia[0] = "Ewa";
$tablica_ucznia[1] = "Kowalska";
$tablica_ucznia[2] = "17-03-1996";

$tablica_klasy[2] = $tablica_ucznia;

echo $tablica_klasy[1][0]; // powinno wyświetlić
```

```
                    // Krzysiek
?>
```

Zasada odwoływania się do indeksów jest podobna jak w przypadku jednowymiarowych tablic. Trzeba pamiętać, że wymiar tablicy liczymy od prawej. W naszym przykładzie indeks z prawej strony opisuje dane konkretnego ucznia, z lewej natomiast numer ucznia.

Trzeci wymiar

Można również nieco zmodyfikować przechowywanie danych, zbierając wszystkie tablice klas do jednej tablicy szkoły. Powstanie nam wtedy tablica trójwymiarowa. Idąc dalej tym tropem możemy kilka szkół wrzucić do jednej tablicy, np. szkoły podstawowe, tworząc cztery wymiary. Wtedy do imienia konkretnego ucznia w którejś klasie w jednej ze szkół odwołamy się tablica_szkol[2][3][5][0], gdzie 0 oznacza pierwsze pole w danych ucznia, czyli jego imię. Potrafisz już stosować wielowymiarowe tablice. Wiesz już prawie wszystko o tablicach. Kolejnym tematem są tablice z definiowanymi indeksami. Zapraszam!

Definiowanie indeksu tablicy

Możemy również zdefiniować własne indeksy dla tablic. Np. zamiast pisać $tablica_danych[0], możemy użyć $tablica_danych[‚imie’].
Zobacz poniższy listing:

```
<?php

$tablica_ucznia['imie'] = "Janek";
$tablica_ucznia['nazwisko'] = "Kowalski";
$tablica_ucznia['data_ur'] = "14-10-1995";

$tablica_klasy[0] = $tablica_ucznia;

$tablica_ucznia['imie'] = "Krzysiek";
$tablica_ucznia['nazwisko'] = "Nowak";
$tablica_ucznia['data_ur'] = "24-12-1994";

$tablica_klasy[1] = $tablica_ucznia;

$tablica_ucznia['imie'] = "Ewa";
$tablica_ucznia['nazwisko'] = "Kowalska";
$tablica_ucznia['data_ur'] = "17-03-1996";

$tablica_klasy[2] = $tablica_ucznia;

echo $tablica_klasy[1]['imie']; // powinno wyświetlić
                    // Krzysiek
?>
```

Zestawienie danych wygląda nieco bardziej czytelnie. Sposób indeksowania pozostawiam do Waszego wyboru. Istnieje zupełna dowolność, jak komu wygodniej.

Data i czas

PHP udostępnia nam funkcje pozwalające uzyskać aktualną godzinę, datę i znacznie więcej. Są one bardzo ważne w tworzeniu dynamicznych treści na stronach www. Choćby dla tak trywialnych zastosowań, jak data dodania posta na forum lub data ostatniego logowania. Data złożenie zamówienia w sklepie internetowym jest równie dobrym przykładem.
Innym, bardziej zaawansowanym, byłoby sprawdzanie, czy kliknąłeś w link resetujący hasło w ciągu 30 minut od jego otrzymania. Jeszcze jeden przykład: czy użytkownik nie loguje się akurat w dniu swoich urodzin. Jeśli tak, system mógłby złożyć mu życzenia.
Zastosowań jest dużo, zacznijmy od podstaw. Przedstawię poniżej przykład użycia funkcji date():

```
<?php

$data=date("Y-m-d");
$czas=date("H:i");

echo "Stronę wyświetlono dnia $data o godzinie $czas";

?>
```

Kilka słów wyjaśnień. Funkcja **date** przekształca otrzymane argumenty na ciąg znaków. Wszystko co nie jest formatem daty (u nas „-” oraz „:”) pozostaje niezmienione. W naszym przykładzie Y oznacza rok w formacie czterocyfrowym, m – miesiąc dwucyfrowy, d – dzień miesiąca, również dwucyfrowy. H oznacza godzinę, a i minutę. Kompletną specyfikację dostępnych formatów dla funkcji **date** po polsku znajdziecie tutaj – PHP.net Date Manual.

Data wcześniejsza lub późniejsza

Funkcja **date** może również wyświetlić datę inną niż obecna. Żeby to osiągnąć należy dodać jej dodatkowy argument w postaci funkcji mktime(). Zwraca ona ilość sekund, które upłynęły od 1 stycznia 1970 roku do momentu podanego jako argument.
Zobaczmy przykład:

```
<?php

$data=date("Y-m-d, H:i", mktime (0,0,0,10,15,1985));

echo $data;

?>
```

Funkcja **mktime** przyjmuje aż 6 argumentów. Licząc od lewej: godzina, minuta, sekunda, miesiąc, dzień, rok. Powyższy przykład wyświetli datę 15 października 1985 roku. Istnieją jeszcze dwie ważne funkcje traktujące o upływie czasu **– time()** i **microtime()**. Obie zwracają dokładny czas, jaki upłynął od 1 stycznia 1970, z tym, że pierwsza w sekundach, a druga w milisekundach. Przydadzą się przy generowaniu liczb losowych lub przy czasie ładowania strony.

Podsumowanie

Naszym zadaniem będzie napisanie funkcji, która na podstawie wpisanej daty urodzin wyświetli, jaki był to dzień tygodnia. Napiszemy również drugą funkcję, zwracającą ilość dni, które upłynęły od danej daty. Data przekazywana będzie w postaci tablicy ze zdefiniowanymi indeksami.
Napiszmy najpierw plik urodziny.html, który wyświetli formularz umożliwiający wpisanie daty:

```
<html>
<head>
  <title>Wpisz datę urodzenia</title>
</head>
<body>
  <form action="oblicz.php" method="get" >
    Wpisz dzień: <input type="text" name="dzien" /><br/>
    Miesiąc: <input type="text" name="miesiac" /><br/>
    Rok: <input type="text" name="rok" /><br/>
    <input type="submit" value="OK" /><br/>
  </form>
</body>
</html>
```

Nie dzieje się tutaj nic nadzwyczajnego. Zwykły formularz wysyłający datę dalej. Teraz kolej na plik oblicz.php:

```
<?php

function wypisz_dzien_tygodnia($data)
{
  echo date("l", mktime (0,0,0,$data['miesiac'],
  $data['dzien'],$data['rok']));
}

function oblicz_dni($data)
{
 // 60 sekund to 1 minuta, 60 minut to 1 godzina,
 24 godziny to 1 dzień
 $czas = (time() - mktime (0,0,0,$data['miesiac'],
 $data['dzien'],$data['rok']))/60/60/24;
 return $czas;
}

$data['dzien'] = $_GET['dzien'];
$data['miesiac'] = $_GET['miesiac'];
$data['rok'] = $_GET['rok'];

wypisz_dzien_tygodnia($data);

echo oblicz_dni($data);
```

```
?>
```

Przegląd rozwiązania

Przeanalizujmy wspólnie rozwiązanie tego przykładowego problemu. Funkcja **wypisz_dzien_tygodnia** przyjmuje **$data** jako argument**. $data** to zmienna tablicowa, która przechowuje informacje o dniu, miesiącu oraz roku. Format „l" w funkcji **date()** oznacza wyświetlenie dnia tygodnia w języku angielskim. Podsumowując, funkcja **wypisz_dzien_tygodnia($data)** wyświetli dzień tygodnia na podstawie podanej daty.

Kolejna funkcja – **oblicz_dni($data)**, również przyjmuje jako argument **$data**. Korzysta z funkcji **time(),** zwracającej liczbę sekund, które upłynęły od 1. stycznia 1970 roku. **Mktime()** przyjmuje jako argument podaną datę i zwraca liczbę sekund od 1. stycznia 1970 do tej daty. Wynika z tego, że gdy odejmiemy jedną wartość od drugiej otrzymamy szukaną liczbę sekund – od daty do dnia dzisiejszego. Pozostaje tylko podzielić wartość przez 60/60/24, aby zamienić sekundy na dni. Tym razem funkcja zwraca wartość.

Reszta kodu wydaje się być oczywista. Przypisanie danych wysłanych z formularza do zmiennej tablicowej, wywołanie dwóch funkcji i zakończenie pliku. Wartość zwracana przez funkcję **oblicz_dni($data)** może zostać od razu przekazana do funkcji echo, która spowoduje wyświetlenie liczby dni.

Ćwiczenia

- Zmodyfikuj funkcję **wypisz_dzien_tygodnia**, by wyświetlał nazwy dni tygodnia po polsku. Użyj formatu „w" funkcji date() i instrukcji warunkowej switch.
- Napisz skrypt sprawdzający pełnoletność użytkownika. Odwiedzający wpisuje swoją datę urodzenia i w zależności od obliczonego wieku wyświetla się stosowny komunikat.
- Stwórz funkcję, która zapisze do tablicy dni tygodnia dziesięciu kolejnych dni matki (26.05), a następnie zwróci tę tablicę