

# PHP – funkcje, szablony

Beata Pańczyk - PHP (Wykład 4)

# Plan wykładu

- Definicja funkcji
- Sposoby przekazywania parametrów funkcji
- Zasięg zmiennych
- Funkcja require i include
- Pliki zdalne
- Szablony stron
- Pliki i katalogi

# Funkcje w PHP

- Nazwa funkcji:
  - nie może rozpoczynać się cyfrą
  - może zawierać jedynie litery, cyfry i kreski podkreślenia
  - umownie małe litery
- Funkcja może zawierać:
  - definicje kolejnych funkcji
  - definicje klas
- Wszystkie funkcje (i klasy) w PHP mają zasięg globalny – mogą być wołane z zewnątrz nawet jeśli są zdefiniowane wewnątrz innej funkcji
- Brak przeciążania nazw funkcji

# Definicja funkcji

```
function nazwa_funkcji($arg1, $arg2, ...)  
{  
    instrukcje;  
    return $wynik; //nie musi występować  
    //return; - wstrzymuje wykonanie funkcji  
}
```

- Dopuszczalna jest dowolna liczba argumentów funkcji
- Dopuszczalne są wartości domniemane parametrów
- Domyślnie parametry przekazywane przez wartość
- Możliwość przekazywania parametrów przez referencje
- Możliwość przekazywania zmiennej liczby parametrów
- Możliwe są wywołania rekurencyjne funkcji

# Zasięg zmiennych

- **lokalny** - zmienne zadeklarowane wewnątrz funkcji - **zmienne lokalne** (widoczne od miejsca deklaracji do końca funkcji)
- **globalny** - zmienne zadeklarowane na zewnątrz funkcji - **zmienne globalne** (widoczne od miejsca deklaracji do końca pliku, **ale nie wewnątrz funkcji!** )
- słowo kluczowe **global** można zastosować do ustalenia globalnego zasięgu zmiennej zdefiniowanej lub stosowanej wewnątrz funkcji
- funkcja `unset($nazwa_zmiennej);` - usunięcie zmiennej

# Przykład 1a - zasięg zmiennych

<?

```
function fun()
```

```
{
```

```
    echo "W funkcji: \$n=".$n."<br />";
```

```
    $n=5;
```

```
    echo "W funkcji: \$n=".$n."<br />";
```

```
}
```

```
$n=10;
```

```
fun();
```

```
echo "Na zewnątrz funkcji: \$n=".$n."<br />";
```

?>

W funkcji: \$n=

W funkcji: \$n=5

Na zewnątrz funkcji: \$n=10

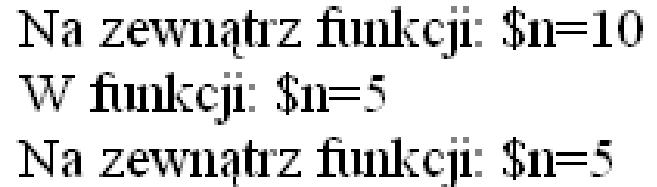
# Przykład 1b - zasięg zmiennych

<?

```
function fun()
{ global $n;
  $n=5;
  echo "W funkcji: \$n=".$n."<br />";
}
$n=10;
echo "Na zewnątrz funkcji: \$n=".$n."<BR>";
fun();
echo "Na zewnątrz funkcji: \$n=".$n."<BR>";
```

?>

- **UWAGA!** - miejsce deklaracji funkcji nie ma znaczenia - ważne jest miejsce wywołania funkcji



Na zewnątrz funkcji: \$n=10  
W funkcji: \$n=5  
Na zewnątrz funkcji: \$n=5

# Przekazywanie parametrów

- **przez wartość** - w bloku funkcji tworzona jest kopia oryginału, można ją dowolnie modyfikować, ale wartość oryginalnej zmiennej poza funkcją pozostaje niezmienną, np.:

```
<? function dodaj($a,$b) { $a=$a+$b;}
```

```
$a=10; dodaj($a,5); echo $a;
```

```
?>
```

- **przez referencję** - funkcja otrzymuje referencję (odniesienie) do oryginału, każda modyfikacja referencji dotyczy również oryginału, np.:

```
<? function dodaj(&$a,$b=5) { $a=$a+$b;}
```

```
$a=10; dodaj($a); echo $a;
```

```
?>
```



# Przykład 2 - zastosowanie funkcji

- Skrypt **form3.php** pyta użytkownika o informacje, sprawdza czy wymagane pola zostały wypełnione i wyświetla ekran zatwierdzający.
- Wykorzystano funkcje:
  - drukuj\_form (wydruk formularza z uwzględnieniem wypełnionych już pól),
  - sprawdz\_form (sprawdzenie danych wprowadzonych przez użytkownika)
  - zatwierdz\_form (wyświetlenie informacji wprowadzonych przez użytkownika).

# Przykład 2 - form3.php

```
<?php  /*deklaracje funkcji*/  
function drukuj_form($imie, $nazw, $tel) { ?>  
<form method="post" action="http://localhost/form3.php">  
    Nazwisko *: <input type="text" name="nazw" value="  
        <?php print $nazw ?> "><br />  
    Imię: <input type="text" name="imie" value="<? print $imie ?>"><br />  
    Telefon *:</b><input type="text" name="tel" size=10 value="  
        <? print $tel ?>"><br />  
    <input type="submit" name="submit" value="Wyslij zamówienie">  
    <input type="reset" value="Anuluj zamówienie">  
</form>  
<?php  }           //koniec funkcji drukuj_form
```

# Przykład 2 - form3.php

## c.d.

```
function sprawdz_form($imie,$nazw,$tel)
{
    if (!$nazw||!$tel)
    { print("<h3> Opuściłeś niektóre pola wymagające wypełnienia!<h3>");
      if (!$nazw) { print("Prosimy wpisać nazwisko <br />"); }
      if (!$tel)   { print("Prosimy wpisać numer telefonu.<br />");}
      drukuj_form($imie, $nazw, $tel);
    }
    else zatwierdz_form($imie, $nazw, $tel);
} //koniec funkcji sprawdz_form
```

# Przykład 2 - form3.php

## c.d.

```
function zatwierdz_form($imie, $nazw, $tel)
{
    ?>
    <h2> Dziękujemy! Oto informacje, które do nas wysłałeś:</h2>
    Informacje kontaktowe:<br />
    <?  print("$nazw $imie <br /> tel: $tel <br />");
} //koniec funkcji zatwierdz_form

/*początek programu głównego: */

if (!isset($_POST['submit']))
{
    ?>  <h3> Prosimy o wpisanie swoich danych</h3>
    Pola ze znakiem "<b>*</b>" wymagają wypełnienia.<p>
    <?php drukuj_form("", "", "");    }
else sprawdz_form($_POST['imie'], $_POST['nazw'], $_POST['tel']);
?>
```

# Przykład 2 - widok form3.php

**Prosimy o wpisanie swoich danych**

Pola ze znakiem "\*" wymagają wypełnienia.

Nazwisko \*:

Imię:

Telefon \*:

# Przykład 2 – wynik działania

**Opuściłeś niektóre pola wymagające wypełnienia!**

**Prosimy wpisać numer telefonu.**

**Nazwisko \*:**

**Imię:**

**Telefon \*:**

**Dziękujemy! Oto informacje,  
które do nas wysłałeś:**

**Informacje kontaktowe:**

Wiśniewski Andrzej

tel:666666

# Domniemane parametry tablicowe

- PHP dopuszcza stosowanie tablic i wartości specjalnej NULL jako wartości domniemane
- Wartości domyślne muszą być wyrażeniem stałym
- np.:

```
<?php
```

```
function zrob_kawe($rodzaj=array("cappucino"),  
    $ekspres=NULL)  
{ $jak=is_null($ekspres)?"ręcznie":$ekspres;  
  return "Parzę kawę ".join(", ",$rodzaj)." $jak.<br />";  
}  
echo zrob_kawe();  
echo zrob_kawe(array("cappucino","lavazza","latte"),"z  
    ekspresu").
```

```
?>
```

Parzę kawę cappucino ręcznie.

Parzę kawę cappucino, lavazza, latte z ekspresu.

# Przekazywanie przez funkcję wielu wartości

- Funkcja nie może zwracać wielu wartości, ale może zwracać tablicę wartości
- Np.:

```
<?php
function liczby()
{
    return array (0, 1, 2);
}
list ($zero, $jeden, $dwa) = liczby();
?>
```



# wynik funkcji przekazywany przez referencję

```
<?php
    function &przekaz_referencje()
    {
        //oblicz $ref;
        return $ref;
    }

    $nowa_ref =& przekaz_referencje();
?>
```

# Funkcje require i include

- Dołączają i wykonują wskazany plik
- **require()** działa analogicznie do **include()**, tylko w razie błędu generuje błąd (fatal error) **E\_COMPILE\_ERROR** i zatrzymuje działanie skryptu podczas gdy **include()** zgłasza jedynie komunikat **E\_WARNING** (warning), co pozwala skryptowi na dalsze działanie
- Jeśli w pliku php.ini jest włączona opcja `allow_url_include = On` - można dołączać pliki wskazując ich URL (poprzez protokół HTTP lub inny)
- Jeśli serwer docelowy interpretuje plik docelowy jako kod PHP, zmienne mogą być przekazane do dołączanego pliku przy użyciu URL tak jak w przypadku HTTP GET. Ale nie jest to zupełnie to samo co w przypadku dołączania zwykłego pliku bowiem skrypt jest uruchamiany na zdalnym serwerze a dopiero rezultat działania skryptu jest dołączany do skryptu lokalnego

# Dołączanie plików

- Dołączane pliki powinny mieć rozszerzenie **.php** i ze względów bezpieczeństwa powinny się znajdować w odrębnym katalogu
- Dowolny kod w pliku dołączanym, który powinien podlegać parsowaniu PHP musi być umieszczony w znacznikach `<?php ... ?>`
- Jeśli dołączany plik nie jest ujęty w znaczniki `<? php ... ?>` jest traktowany jako zwykły tekst lub HTML
- Zastosowanie - tworzenie szablonów stron

# Funkcja require() - działanie

- Plik **tresc.php**:

```
<?php echo "<h4>Witamy na naszej stronie domowej</h4>";  
?>
```

- Plik **glowny.php**:

```
<?php  
echo "<h1>Nagłówek  
strony</h1>";  
require("tresc.php");  
echo "<h6>Stopka</h6>";  
?>
```

## Nagłówek strony

Witamy na naszej stronie domowej

Stopka

# Dołączanie plików poprzez URL

```
<?php
/* Zakładamy, że serwer www.example.com jest
skonfigurowany
* tak aby parsował pliki .php a nie .txt. Zakładamy również,
że
* zmienne $x i $y są dostępne wewnątrz dołączanych
plików. */

// Nie zadziała - file.txt nie jest parsowany przez
// www.example.com jako PHP:
include 'http://www.example.com/file.txt?x=1&y=2';

// Nie zadziała - poszukiwany jest plik 'file.php?x=1&y=2' w
// lokalnym systemie plików:
include 'file.php?x=1&y=2';

// Zadziała:
include 'http://www.example.com/file.php?x=1&y=2';

?>
```

# Pliki zdalne

- Pliki zdalne mogą być przetwarzane na zdalnym serwerze (w zależności od rozszerzenia pliku i tego czy serwer zdalny uruchamia pliki PHP czy nie) – niezależnie od tego pliki te muszą zawierać poprawny skrypt PHP ponieważ są one przetwarzane na serwerze lokalnym (jako dołączone pliki)
- Jeśli plik z serwera zdalnego ma być wykonany i tylko jego wynik ma być pobrany, lepszym rozwiązaniem jest funkcja `readfile()`; w przeciwnym razie należy zwrócić szczególną uwagę na to aby zdalny skrypt zawierał poprawny kod.

# Funkcje `require_once` i `include_once`

- **`require_once()`** – identyczne działanie jak `require()`, poza tym, że PHP sprawdzi czy żądany plik nie został już dołączony i jeśli tak to nie dołączy go ponownie
- **`include_once()`** – j.w.
- Wykorzystywane w przypadku kiedy te same pliki mogą być dołączane i wykonywane więcej niż jeden raz (pozwała to uniknąć problemów związanych z redefiniowaniem funkcji, nadpisywaniem zmiennych itp.)

# Przykład 3 - szablon stron

Plik **glowny.php**:

```
<?    require_once("inc/naglowek.php");  
      require_once("inc/tresc.php");  
      require_once("inc/stopka.php");  
?>
```



# Przykład 3 - naglowek.php

```
<!doctype ...>
<html>
  <head>
    <meta ... />
    <title> Szablonik strony</title>
    <link rel="stylesheet" href="styl.css" type="text/css" />
  </head>
  <body>
    <div id = "kontener" > <!-- kontener główny -->
      <div id = "nag" >
        <!-- TYTUŁ STRONY -->
        <!-- Nawigacja -->
      </div>
```

# Przykład 3 – pozostałe pliki

- plik **tresc.php**:  
    <div id = "tresc" >  
    <! – Główna zawartość strony -->  
    </div>
- plik **stopka.php**:  
    <div id = "stopka" >  
    <! – Zawartość stopki -->  
    </div>  
    </div> <! – koniec kontenera głównego -->  
    </body>  
    </html>

# Operacje plikowe

- `$wynik=rename($stara_nazwa,$nowa_nazwa);`  
`if ($wynik) echo "Nazwa pliku została zmieniona.";`
- `copy()`  
`$wynik=copy($nazwa_pliku1,$nazwa_pliku2);`  
`if ($wynik) echo "Plik został skopiowany.";`
- `unlink()`  
`if (unlink($nazwa_pliku)) echo "Plik został usunięty.";`
- `if (file_exists($nazwa_pliku) echo "Plik $nazwa_pliku istnieje.";`  
`else echo "Plik $nazwa_pliku nie istnieje.";`

# Informacje o pliku: data utworzenia, ostatni dostęp, modyfikacja

- `filetime()` – czas ostatniego dostępu
- `filemtime()` – czas ostatniej modyfikacji
- powyższe funkcje zwracają czas w formacie UNIXa, który należy skonwertować funkcją `date()`

```
<?php
$plik = 'osoby.txt';
if (file_exists($plik))
{ echo "Ostatni dostęp do pliku $plik: " . date("F d Y H:i:s.",
    filetime($plik)); echo "<br />";
  echo "Ostatnia modyfikacja pliku $plik: " . date("d m Y
    H:i:s.",
    filemtime($plik));
} ?>
```

Ostatni dostęp do pliku osoby.txt: November 21 2011 12:21:20.  
Ostatnia modyfikacja pliku osoby.txt: 21 11 2011 13:14:22.

# Szczegółowe informacje o nazwie pliku

- `pathinfo("nazwa_pliku")` – pobiera informację o nazwie pliku i zwraca ją w postaci tablicy asocjacyjnej czterech elementów o kluczach: `dirname`, `basename`, `extension`, `filename`
- np.:  

```
$info_pliku = pathinfo("osoby.txt");  
var_dump($info_pliku);
```

```
array(4) { ["dirname"]=> string(1) "."  
["basename"]=> string(9) "osoby.txt"  
["extension"]=> string(3) "txt"  
["filename"]=> string(5) "osoby" }
```

# Formularz wysyłania plików na serwer

- metoda POST, parametr `enctype="multipart/form-data"` (tzn. że plik będzie wysłany razem ze zwykłymi danymi formularza)
- pole tekstowe typu plikowego (przeglądarka wyświetla okno dialogowe do wyboru pliku z przyciskiem Wybierz lub Przeglądaj)  
`<input type="file" name="plik" >`
- wielkość przesyłanych plików (w `php.ini`):  
`upload_max_filesize = 16M`
- informacje o przesłanych plikach – tablica **`$_FILES`**

# Informacje na temat pliku – 5 elementowa tablica w tablicy \$\_FILES

- Funkcja var\_dump(\$\_FILES) wyświetli np.:

```
array(1) {  
    ["plik"] => array(5)  
    {  
        ["name"] => string(10) "jesien.jpg"  
        ["type"] => string(10) "image/jpeg"  
        ["tmp_name"] => string(24)  
                        "C:\xampp\tmp\phpD9FB.tmp"  
        ["error"] => int(0)  
        ["size"] => int(754344)  
    }  
}
```

# Informacje o pliku

- `name` – oryginalna nazwa pliku
- `type` – typ pliku MIME (jeśli jest znany)
- `tmp_name` – nazwa przekazanego pliku na serwerze
- `error` – informacja o błędzie
- `size` – wielkość pliku w bajtach
- plik po wysłaniu trafia do domyślnego katalogu tymczasowego na serwerze WWW; jeżeli nie nastąpi przeniesienie lub zmiana nazwy pliku, nim skrypt się zakończy - plik zostanie skasowany
- Przeniesienie pliku z serwera - funkcja `move_uploaded_file("plik_uzytkownika", "docelowe_miejsce" )` zwraca `false` lub `true`; sprawdza bezpieczeństwo i wykonuje kopiowanie



# Przykład 4 - formularz wysyłania plików wyslij.html

<h3> Wysyłanie plików do serwera WWW </h3>

<form method="post" action="wyslij.php"  
enctype="multipart/form-data" >

<p>Wybierz plik: <input name="plik" type="file"  
size="50"></p>

<p><input type="submit" name="submit"  
value="Wyślij"></p>

## Wysyłanie plików do serwera WWW

Wybierz plik:

D:\BeataP\Obrazki\jesien.jpg

Przeglądaj...

Wyślij

# Przykład 4 – skrypt wyslij.php

<h3> Wysyłanie pliku ... </h3>

```
<? $tmp_name=$_FILES['plik']['tmp_name'];  
    $name=$_FILES['plik']['name']; $size=$_FILES['plik']  
    ['size'];  
    $path=$_SERVER['DOCUMENT_ROOT']."/../Mojepliki/"  
    $name;  
    if (move_uploaded_file($tmp_name,$path))  
        echo "Wysłano plik: ".$name. ", o rozmiarze: ".$size."  
        bajtów."  
    else "Plik nie został wysłany."  
    echo  
?>
```

**Wysyłanie pliku ...**

Wysłano plik: jesien.jpg, o rozmiarze: 754344 bajtów.

# Restrykcje na wysyłane pliki

<?php

```
if ( (($_FILES["plik"]["type"]=="image/gif")
    || ($_FILES["plik"]["type"] == "image/jpeg")
    || ($_FILES["plik"]["type"] == "image/jpg"))
    && ($_FILES["plik"]["size"] < 1000000))
{ if ($_FILES["plik"]["error"] > 0)
  {   echo "Error: " . $_FILES["plik"]["error"] . "<br />";   }
  else
  {   echo "Przesyłany plik: " . $_FILES["plik"]["name"] . "<br />";
      echo "Typ: " . $_FILES["plik"]["type"] . "<br />";
      echo "Rozmiar: " . ($_FILES["plik"]["size"] / 1024) . " Kb<br />";
      echo "Folder tymczasowy: " . $_FILES["plik"]["tmp_name"];
      // wywołanie funkcji move_uploaded_file
  }
}
else
{
    echo "Błędny plik";
}
?>
```

Przesyłany plik: domek.jpg

Typ: image/jpeg

Rozmiar: 550.509765625 Kb

Folder tymczasowy: C:\xampp\tmp\php5F9A.tmp

# Odczyt praw dostępu do pliku i jego statusu

- jednoargumentowe funkcje `is_readable()`, `is_writeable()`, `is_executable()`, `is_file()`, `is_dir()` zwracają wartość boolean; działają na plikach serwera WWW (na zdalnych nie)
- przy kilkakrotnym wywołaniu funkcji `is_file()` należy wyczyścić buforowane dane funkcją `clearstatecache()`
- `fileowner()` – odczytanie właściciela pliku
- np. sprawdzenie prawa do odczytu:  

```
$nazwa_pliku="c:\Windows\wyslij.html"; //Windows  
$nazwa_pliku="etc/passwd";           //Unix  
if (is_readable($nazwa_pliku)) print file_get_contents($nazwa_pliku);  
else print "Nie masz prawa do odczytu pliku: $nazwa_pliku";
```
- np. sprawdzenie właściciela:  

```
$wlasciciel=fileowner("etc/passwd");  
if ($wlasciciel!=0) print "Ostrzeżenie: właścicielem36  
jest root!"
```

# Przykład 5 – prawa dostępu

```
<? $nazwa_pliku="osoby.txt";  
  if (is_file($nazwa_pliku))  
  { print "Plik $nazwa_pliku istnieje.<br />";  
    if (is_readable($nazwa_pliku))  
      print file_get_contents($nazwa_pliku);  
    else  
      print "Nie masz prawa do odczytu pliku:  
$nazwa_pliku";  
  }  
?>
```

Plik osoby.txt istnieje.

1 Ania Kowalska 2 Magdalena Nowak  
3 Jan Abacki 4 Adam Babacki

# Praca z katalogami

- `opendir()` - otwarcie katalogu do odczytu funkcja zwraca uchwyt katalogu podobnie jak funkcja `fopen` zwraca uchwyt pliku), np.:  
`$kat = opendir( $aktualny_katalog );`
- `readdir($kat)` - odczyt plików z otwartego katalogu; zwraca `false`, kiedy wszystkie pliki zostaną już odczytane; pliki nie są uporządkowane w żaden określony sposób
- `rewinddir($kat)` - powrót do odczytu nazw plików od początku katalogu
- `mkdir($kat, prawa_dostępu)` – zwraca `true` lub `false`
- `rmdir($kat)` – usuwa pusty katalog
- `closedir($kat)`

# Przykład 6 - przeglądanie katalogu

```
<h3> Zawartość katalogu:<br />
```

```
<?
```

```
$katalog=$_SERVER['DOCUMENT_ROOT']."/../Mojepliki/";  
$kat=@opendir($katalog) or die("Nie można otworzyć  
katalogu");
```

```
echo "$katalog</h3>";  
while ($plik=readdir($ka  
    echo "$plik<br />";  
closedir($kat);
```

```
?>
```

**Zawartość katalogu:**

**C:/xampp/htdocs/../Mojepliki/**

.

..

domek.jpg

jesien.jpg

liczba.txt

muchomor.jpg

# Przykład 6a - przeglądanie katalogu z możliwością otwarcia/pobrania pliku

Modyfikacja pętli while z przykładu 6:

```
while ($plik=readdir($kat))  
    echo "<a href='$katalog/$plik'>$plik</a><br/>";
```

**Zawartość katalogu:**

**C:/xampp/htdocs/../../Mojepliki/**

.

..

[domek.jpg](#)

[jesien.jpg](#)

[liczba.txt](#)

[muchomor.jpg](#)



# Przykład 6b - wyświetlanie informacji o pliku

Modyfikacja pętli while z przykładu 6a:

```
while ($plik=readdir($kat))  
    echo "<a href='wlasciwosci_pliku.php?plik=$plik&&katalog=$katalog' >  
    $plik</a><br />";
```

**Zawartość katalogu:**

**C:/xampp/htdocs/../../Mojepliki/**

=

=

[domek.jpg](#)

[jesien.jpg](#)

[liczba.txt](#)

[muhomoc.jpg](#)

<http://localhost/Mojepliki/>

**WŁAŚCIWOŚCI PLIKU: jesien.jpg**

Katalog: C:/xampp/htdocs/../../Mojepliki/

Utworzony: 27 November 2011 21:19

Zmodyfikowany: 27 November 2011 21:19

Typ pliku: file

Rozmiar pliku: 754344

# Przykład 6b - wlasosci\_pliku.php

<?

```
$katalog=$_GET['katalog'];  
$name=$_GET['plik']; $plik="$katalog/".$name";  
$utworz=filetime($plik); $modyf=filemtime($plik);  
$data_u=date("j F Y H:i", $utworz);  
$data_m=date("j F Y H:i", $modyf);  
echo "<h4>WŁAŚCIWOŚCI PLIKU: $name </h4>";  
echo "Katalog: $katalog <br >";  
echo "Utworzony: $data_u <br/>";  
echo "Zmodyfikowany: $data_m <br/>";  
echo "Typ pliku: ".filetype($plik)."<br/>";  
echo "Rozmiar pliku: ".filesize($plik)."<br/>";
```

?>

# Przykład 7 – pliki zdalne

```
<?php
    echo "<h2>Strona domowa</h2>";
    //Połączenie z URL i odczytanie informacji:
    $url="http://cs.pollub.pl/~beatap/";
    $wp=@fopen($url,"r") or die("Otwarcie url niemożliwe");
    $zawartosc=strip_tags(fread($wp,100000));
    echo $zawartosc; fclose($wp);
```

?>

