12. Funkce v PHP

Funkce v PHP dělíme na dvě základní skupiny – vestavěné a uživatelské.

Vestavěné funkce

Samotné PHP obsahuje doslova tisíce funkcí, které můžete libovolně používat ve svých skriptech. Těmto tedy říkáme vestavěné funkce. Jedním ze základních příkladů je funkce phpinfo:

```
<?php
phpinfo();
?>
```

Tato vestavěná funkce nám například vypíše kompletní informace o PHP serveru.

Funkci můžeme přidat také určité **parametry**. Ty jsou umístěny v závorkách za názvem funkce. Příklad:

```
<?php
print('ahoj');
// V tomto případě pouze vypíše ahoj.
?>
```

Dále může funkce tzv. něco vracet (například vypisovat).

```
<?php
$cislo = pi();
// Proměnná $cislo bude obsahovat číslo pí

echo $cislo;
// Výpis čísla pí

echo '<br />';
// Zařídí, aby se další vypsalo na nový řádek

echo pi();
// Číslo pí je možné také vypsat přímo
?>
```

Nebo příklad z funkcí max:

```
<?php
$cislo = max(4,1);
// Proměnná $cislo bude obsahovat největší číslo z čísel 4 a 1.
echo $cislo;
// Výpis hodnoty proměnné $cislo
echo '<br>';
// Zařídí, aby se další číslo vypsalo na nový řádek. I tuto funkci můžeme vypsat takto:
```

```
echo max(4,1);
// Vypíše největší číslo z čísel 4 a 1
?>
```

Příklad využití vestavěné funkce náhodného čísla – mt_rand (elektronická verze házecí kostky)

Funkce mt_rand vrací náhodné číslo. Má 2 parametry, a to minimální a maximální číslo z rozsahu, který má vrátit. V tomto případě 1 až 6.

```
<?php
 $nahodne_cislo = mt_rand(1,6);
  switch ($nahodne_cislo) {
  case '1':
   echo 'Padlo číslo 1';
   break;
  case '2':
   echo 'Padlo číslo 2';
   break;
  case '3':
   echo 'Padlo číslo 3';
   break;
  case '4':
   echo 'Padlo číslo 4';
   break;
  case '5':
   echo 'Padlo číslo 5';
   break;
  case '6':
   echo 'Padlo číslo 6';
   break;
 }
?>
```

Seznam všech vestavěných funkcí PHP lze najít v oficiálním manuálu PHP – http://php.net nebo např zde – http://php.baraja.cz/

Uživatelské funkce

Pro vytváření uživatelských neboli vlastních funkcí se používá klíčové slovo funkcion. Za klíčovým slovem funkcion následuje název (jméno) funkce, které si sami zvolíme. Zjednodušeně to vypadá následovně:

```
<?php
function jmeno_funkce()
{
telo_funkce;
}
?>
```

Nyní si zkusíme konkrétní jednoduchý příklad, kde se pouze vypíše pozdrav Ahoj:

```
<?php
function vypis()
// Zde jsme vytvořili funkci s názvem vypis. Dále následuje její tělo. V
tomto případě je to pouze příkaz k vypsání slova Ahoj
{
echo 'Ahoj';
}
vypis();
// Zde voláme funkci s názvem vypis. V tomto případě provede vypsání slova
Ahoj.
?>
```

Uživatelské funkce s parametry

Další příklad ukazuje složitější funkci s parametry:

```
<?php
// Deklarace a definice funkce:
function TretiMocnina($Cislo)
{
$Vysledek = $Cislo * $Cislo * $Cislo;
return $Vysledek;
}
// Volání funkce TretiMocnina():
echo TretiMocnina(5); // Vypíše: 125
?>
```

Nejprve je vytvořena funkce s názvem TretiMocnina. Ta obsahuje parametr, který v tomto případě tvoří novou proměnnou s názvem $Cislo. V tělu funkce následuje operace, kdy vytvoříme novou proměnnou s názvem $Vysledek. Tomu je přiřazena operace, která násobí třikrát za sebou proměnnou $Cislo (protože chceme tvořit třetí mocninu, což je trojnásobek). Na dalším řádku máme slůvko returt což je příkaz, který nám vrací hodnotu proměnné $Vysledek. Následuje příkaz echo, který vypíše hodnotu proměnné TretiMocnina s parametrem 5. Neboli do proměnné $Cislo dosadíme tímto číslo 5, které se následně třikrát po sobě vynásobí a vytvoří tedy třetí mocninu.

Další příklad ukazuje, že funkci můžeme ve skriptu použít, kolikrát chceme. Což se velmi často používá:

```
<?php
// Deklarace a definice funkce:
function TretiMocnina($Cislo)
{
$Vysledek = $Cislo * $Cislo * $Cislo;
return $Vysledek;
}
// Volání funkce TretiMocnina():
echo TretiMocnina(5); // Vypíše: 125
?>
```

Další ukázka uživatelské funkce (zdroj http://php.vrana.cz):

```
<?php
// Vrácení českého názvu měsíce

function cesky_mesic($mesic) {
    static $nazvy = array(1 => 'leden', 'únor', 'březen', 'duben', 'květen',
'červen', 'červenec', 'srpen', 'září', 'říjen', 'listopad', 'prosinec');
    return $nazvy[$mesic];
}
echo cesky_mesic(date("n")) . "\n";

// Vrácení českého názvu dne v týdnu
function cesky_den($den) {
    static $nazvy = array('neděle', 'pondělí', 'úterý', 'středa', 'čtvrtek',
'pátek', 'sobota');
    return $nazvy[$den];
}
echo cesky_den(date("w")) . "\n";
?>
```

Pozn.: znaky **\n se v PHP používají pro odřádkování v kódu. Nikoliv ve webové stránce! Pro tu se musí použít klasické
**

Globální a lokální proměnné

Jakákoliv proměnná, která je použita mimo jakoukoliv funkci, se v PHP nazývá globální proměnná. Zato ta, která je použita uvnitř funkce se nazývá lokální proměnná. Tyto proměnné se navzájem nijak nemíchají a jsou jakoby oddělené. Příklad:

```
<?php
$x = 4;
// Globální proměnná $x nastavena na 4.
function vypis_x()
{
echo $x;
// Tohle je pokus o výpis lokální proměnné $x.
}
echo vypis_x();
?>
```

Po spuštění dostanete asi toto chybové hlášení:

Notice: Undefined variable: x in C:\wamp\www\index.php on line 7

Jde o to, že uvnitř funkce s názvem vypis_x se pokoušíme příkazem echo vypsat proměnnou $x. Takováto proměnná ale uvnitř této funkce neexistuje. Nachází se totiž mimo tuto funkci, tak že ji tento příkaz echo uvnitř funkce marně zkouší vypsat. Proměnná $x sice ve skriptu existuje, ale je je to globální proměnná, na kterou „zevnitř“ funkce nevidíme.

Poznámky k funkcím

- Funkce **muže mít více vstupních parametrů**. Ty se pak oddělují čárkou.
- Funkce v PHP **mohou předávat parametry odkazem**. To se provede tak, že před název proměnné v hlavičce funkce se uvede **ampersand (&)**
- **V těle funkce mohou být definovány proměnné s klíčovým slovem *static***. Hodnotu takových proměnných si PHP mezi jednotlivými voláními funkce pamatuje.
- Funkce v PHP **mohou být rekurzivní. To znamená, že funkce muže volat sebe samu**.
- Funkce **nemůže vracet více než jeden výstupní parametr**. Ale muže vracet pole, takže se to dá obejít.
- Jedna **uživatelská funkce muže volat jinou. Na pořadí, v jakém jsou uvedeny ve skriptu, přitom nezáleží**. Volání funkce muže být uvedeno dokonce již dříve než definice funkce samotné.