Функции в PHP

В тази тема:

1. Функции дефинирани от потребителя
2. Аргументи на функция
3. Връщане на стойности

1. Функции дефинирани от потребителя

Синтаксис за дефиниране на функция:

```
function func_name($arg_1, $arg_2, /* ..., */ $arg_n)
{
   ……..;
   ……;
   return $result;
}
```


Правила за използване на функциите

- Използван код във функциите – всякакъв валиден код на PHP, дори дефиниция на др.функции или класове.
- Валидни имена на функции – спазват правилата на другите етикети в PHP. Започва с буква или подчертавка, следвана от букви, цифри или подчертавки.
- Не е необходимо функциите да бъдат дефинирани преди да бъдат използвани, освен в случаите, когато функцията е дефинирана условно.

Условни функции

```
$create_order = true;

// Не може да се извика new_order(), защото тя не съществува
// но може да се извика to_new_order()
to_new_order();

If ($create_order ) {
  function new_order()
  {
     // Функцията не съществува докато изпълнението на програмата
     // не стигне до нея
  }
}
```



```
// Сега може да се извика new_order(), защото $create_order е
// оценена на истина

If ($create_order ) new_order();

function to_new_order()
{
   // функцията съществува от стартирането на програмата
}
```


Функция във функция

```
function new_car()
{
        function new_model()
        {
                // Функцията не съществува, докато не се
                // извика new_car
        }
}
```



```
// Не може да извикаме new_model(), защото
// все още не съществува

new_car()

// Сега може да извикаме new_model()
```

Рекурсивно извикване на функции

- Възможно е рекурсивно извикване на функция.
- Препоръчва се да се избягва рекурсивно извикване с повече от 100-200 нива на рекурсия, защото може да доведе до смачкване на стека, което да предизвика прекратяване на скрипта.

```
function recursion($a)
{
   if ($a < 20) {
      echo "$a\n";
      recursion($a + 1);
   }
}
```


2. Аргументи на функция

- Функцията може да получава параметри под формата на списък от аргументи.
- Аргументите могат да се предават по:
  - **стойност**
  - по **референция**
  - **стойности по подразбиране**
- Поддържат се списъци с документи с променлива дължина

Предаване на аргументи по стойност

- Ако стойността на аргумента се промени вътре във функцията, това няма да промени съответната променлива извън нея.

primer-6-01.php

```
$new_price = 10.20;

function calc_new_price($new_price)
{
        $new_price = $new_price + 1;
        echo 'Новата цена е '.$new_price."<br>";          // 11.20
}

calc_new_price($new_price);

echo 'Стойността на $new_price е '."$new_price";          //10.20

?>
```


Предаване на аргументи по референция

- Аргументите предаване по референция разрешават на функцията да промени стойността на предадената променлива
- Предаването на аргумент по референция се извършва, като той се предхожда от амперсанд (&) пред името в дефиницията на функцията

primer-6-02.php

```
$new_price = 10.20;

function calc_new_price(&$new_price)
{
        $new_price = $new_price + 1;
        echo 'Новата цена е '.$new_price."<br>";                // 11.20
}

calc_new_price($new_price);

echo 'Стойността на $new_price е '."$new_price";                //11.20

?>
```


Използване на аргументи по подразбиране

- Функция може да дефинира стойности по подразбиране за скаларни аргументи.

```
function makecoffee($type = "капучино")
{
   return "Приготвяне на чаша $type.\n";
}
echo makecoffee();                  // Приготвяне на чаша капучино.
echo makecoffee(null);              // Приготвяне на чаша .
echo makecoffee("еспресо");         // Приготвяне на чаша еспресо.
```


Променлива дължина на списъка с аргументи

- Списък с аргументи с променлива дължина може да се използва в потребителските функции.
- Обработката на аргументите се извършва с:

int **func_num_args ()**
Връща броя на аргументите изпращани към функция

mixed **func_get_arg (int $arg_num)**
Връща специфичен аргумент чрез посочения индекс (започва от 0)

array **func_get_args ()**
Връща масив със списъка на аргументи

primer-6-03.php

```
new_car("Иванов","Рено","Меган");
{
   $num_args = func_num_args();
   echo "Брой аргументи: $num_args"."<br>";                    //3

   $second_arg = func_get_arg(1);
         echo "Вторият аргумент е:  $second_arg"."<br>";        // Рено

   $arg_list = func_get_args();
   for ($i = 0; $i < $num_args; $i++) {
      echo "Аргумент $i е: " . $arg_list[$i] . "<br />";
   }

function new_car()
```

3. Връщане на стойности

- Извършва се с **return**, който е незадължителен.
- Може да се връщат всякакви типове, включително списъци и обекти.
- Може да се върне само една стойност;
- Изпълнението на return спира незабавно изпълнението и предава контрола на реда, от който е извикана.


```
function square($num)
{
   return $num * $num;
}
echo square(4);   // 16
$result2 = square(8);       // $result2 ще получи стойност 64
```


Връщане на масив, с цел получаване на множество стойности

```
function odd_digits()
{
   return array (1, 3,  5);
}

list ($one, $three, $five) =  odd_digits();
```


Връщане на референция

- Използва се референтния оператор &, както при декларирането на функцията, така и при присвояването на връщаната стойност на променливата.

```
function &returns_reference()
{
   return $someref;
}

$newref =& returns_reference();
```

Използван източник:

- Ръководство по PHP: Справочник на езика. http://php.net/manual/bg/langref.php