PHP'DE FONKSİYONLAR

Fonksiyonlar genel olarak birden fazla tekrarlanan işlemlerin tek seferde yapılması için kullanılmaktadır.

Birden çok yapılacak işlem için fonksiyon tek seferde yazılır. Kullanılacak yerlerde bu fonksiyonlar çağrılır.

Şu zamana kadar birçok fonksiyon gördük aslında , şimdi ise yapı olarak fonksiyonu kullanmayı öğreneceğiz.

```
<?php
function topla($sayi1,$sayi2)
{
        echo $sayi1+$sayi2;
}
topla(100,437);
?>
```

---

fonksiyon2.php

```
<?php
function bolme($sayi1,$sayi2){
        if($sayi1==0 || $sayi2==0)
        {
   return 0;
        }else{
                return ($sayi1/$sayi2);
        }
}
echo bolme(4,0);
?>
```

Hazırlanan fonksiyonların bazı durumlarda geriye değer döndürmesi gerekir,bunun için **return** fonksiyonu kullanılır.

Parametreli fonksiyonlarda eksik girilirse hata alınır.

Bazı durumlarda ise parametreler isteğe bağlı olabilir, fonksiyonun yapacağı işe bağlı olarak bir veya birden fazla parametreyi zorunlu olmaktan çıkarabiliriz.

Buna varsayılan parametre denir.

varsayılan.php

```
<?php
function h ($mesaj,$s=1){
	return "<h$s>Mesaj</h1>";
}
echo h("Merhaba");
echo h("Merhaba",2);
echo h("Merhaba",3);
//ikinci parametre zorunlu olmaktan çıkmıştır.
?>
```

---

varsayılan2.php

```
<?php
function form($action="",$method="post"){
	return "<form action='$action' method='$method'>";
}
function form_close(){
	return "</form>";
}
function input($name,$aciklama="",$type="text"){
	return "$aciklama<input type='$type' name='$name' /><br>";
}
echo form("giris.php","get");
echo input("ad","Adınızı yazın");
echo input("mail","Eposta yazın");
```

```
echo input("sifre","Şifre yazın");
echo form_close();
?>
```

GLOBAL DEĞİŞKEN

Global ifadeleriyle dışarıdaki bir değişkeni fonksiyon içinde kullanabiliriz.

Birden fazla değişkeni araya virgül koyarak tanımlayabiliriz.

global.php

```
<?php
$b = 2;
function toplam($a)
{
	return $a+$b;
}
echo toplam(5);    //Sonuç 5 olur çünkü fonk.5 değişkenini görmez.
?>

<?php

/*$b = 2;
function toplam($a)
{
	global $b;
	return $a+$b;
}
echo toplam(5);   //Sonuç artık 7 dir.  */

?>
<?php
/*$a = 5;
```

```
$b = 10;
function topla(){
	global $a,$b;
	return $a+$b;
}
echo topla();  //birden fazla değeri virgül ile girebiliriz.*/
?>
```

---

fonksiyon_dizi.php

```
<?php
function dizi(){
	return array("Galatasaray","Fenerbahçe","Beşiktaş");
}
list ($a,$b,$c) = dizi();
echo "$a sarı kırmızı, $b sarı lacivert ve $c de siyah beyazdır";
?>
<?php
function dizi2(){
	$dizi2 = array("Gazete","Kitap","Bilgisayar");
	return $dizi2;
}
$degisken=dizi2();
echo $degisken[1];
?>
<?php
function yeni(){
	return array("a" =>"Aslan" , "b" => "Balık" , "y" => "Yengeç" );
}
$dizi3 = yeni();
echo $dizi3['a'];		?>
```

FONKSİYONLARDA REFERANS ÖZELLİĞİ

Referans özelliği bir değişkenin bir nesnenin birebir aynı özelliklerine ve değerlerine ulaşmak için kullanılan yöntemin adıdır.

PHP'de & işaretiyle tanımlanır.

Değişkenlerde,Fonksiyonlarda,dizilerde,nesnelerde kullanılabilir.

referans.php

```
<?php
$a = 'Bir cümle yazalım';
$b = &$a;
echo $b;
echo $a;
?>
<?php
$a = 'Bir cümle yazalım';
$b = &$a;
$b = "ikinci bir cümle.";   //b nin değerinin değişmesi a yı da değiştirir.
echo $b;
echo $a;
?>
```

İÇİÇE FONKSİYON

iifonksiyon.php

```
<?php
function toplama($a,$b){
        function carpma($a,$b){
                echo "çarpma sonucu : ".($a*$b);
        }
        echo "toplama sonucu : " . ($a+$b),"<br>";
}
toplama(33,67);
carpma(12,12);
```

```
?>
<?php
/*function toplama($c,$d){
	function carpma($c,$d){
		echo "çarpma sonucu : ".($c*$d);
	}
	echo "toplama sonucu : " . ($c+$d),"<br>";
	carpma($c,$d);
}
toplama(3,4);*/
?>
```

RECURSIVE-KENDİ KENDİNİ ÇAĞIRAN FONKSİYON /ÖZYİNELİ-TEKRARLANAN

Fonksiyonun içinde işlem bitene kadar, tekrar tekrar o fonksiyonu çağırabiliriz. Bu işleme ÖZYİNELİ fonksiyon (recursive function) denir.

Tasarımı kolaylaştırır; ancak, fonksiyon çağırma sayısı ve parametre aktarımı artacağı için bellek alanı gereksinimi de artar. Çünkü her fonksiyonun çağrılmasında ve parametrelerin aktarılmasında **yığın (stack)** olarak adlandırılan bellek alanı kullanılmaktadır.

Recursive fonksiyonlar, **permütasyon, hızlı sıralama, fibonnaci** sayılarının bulunması gibi algoritmaların gerçekleştirilmesi için son derece uygundur ve çok hızlıdır.

Recursive fonksiyonların çalışıp çalışmayacağına emin olmamız için ilk önce düşünmemiz gereken şey; fonksiyon her çağrıldığında yapılan işin bir önceki çağrılmaya göre azaldığı olmalı. Yani, recursive fonksiyonlar **döngüler gibi çalıştığından** dolayı yapılan işin bir sınırı olmalı ve her bir adımdan sonra fonksiyonumuzun bu sınıra yaklaştığına emin olmalıyız.

recursive.php

```
<?php
function faktoriyel($n)
{
	if($n==0)
		return 1;
	else
		return $n * faktoriyel($n-1);
}
```

```
echo faktoriyel(6);
?>
```

---

recursive2.php

```
<?php

//recursive..

function recursive($num){

   echo $num, '<br>';

   if($num < 50){

      return recursive($num + 1);
   }
}

$startNum = 1;

//çağıralım fonk.
recursive($startNum);
?>
```

---

function_exists() bir fonksiyonun varlığını test eder.

exist.php

```
<?php
function ornek(){
	return false;
}
echo function_exists("ornek") ? "Fonskiyon Var" : "YOOOKK" ;
echo function_exists("ornek2") ? "Fonskiyon Var" : "YOOOKK" ;
?>
```

FONKSİYON ÇAĞIRMA

call_user_func() ile bir fonksiyonu çağırıp,kullanabiliriz.

call.php

```
<?php
function dersler($a){
	print_r($a);
}
$p = array('Ders1' => 'Programlama' , 'Ders2' => 'Algoritmalar');
call_user_func('dersler',$p);

//call_user_func_array ile elemanları dizi olarak atayıp çağırabiliriz.

function dersler2($anahtar,$deger){
	print $anahtar;
	print $deger;
}
call_user_func_array('dersler2',array('DersA','Bilgisayar Ağları'));
?>
```

---

new.php

```
<?php
function dersler(){
	$sayisi = func_num_args(); //atanan argüman sayısını verir.

	echo "Toplam argüman sayısı:" .$sayisi;
	echo "<br>";
	echo "2. sıra : " .func_get_arg(2); //tek argümanı kullanır.
	echo "<br>";
	$argumanlar = func_get_args();   //listesini verir

	for($i=0; $i<$sayisi; $i++){
		echo "Argüman $i:" .$argumanlar[$i]."<br>\n";
	}
}
dersler('Web','Matematik','Ağlar','Siber Güvenlik');
?>
```

---

ortalama.php

```
<?php
function ortalama(){
	$parametre_sayisi = func_num_args();
	$parametreler = func_get_args();
	return (array_sum($parametreler) / $parametre_sayisi);
}
echo ortalama(10,20,30,60);
?>
```

ANONİM FONKSİYONLAR

Fonksiyona isim vermeden , fonksiyon gövdesiyle yapılan kodlamaya anonim fonksiyon denir.

anonim.php

```
<?php

$mesaj = function($anonim)

{

	printf("Merhaba %s",$anonim);

};

$mesaj('Dünya');

$mesaj ('PHP');

?>
```

SINIF KAVRAMI – CLASS

Fonksiyonları gruplandırarak bir arada kullanmamızı sağlar. Aynı amacı gerçekleştiren fonksiyonların biraraya gelmesidir.

Bu bütünlüğü oluşturan fonksiyonların tümüne **metot** denir. Fonksiyonlardan yani metotlardan elde edilenlere ise **nesne** denir. Kısaca sınıflardan **object/nesne** türü veriler elde edilir.

class.php

```
 <?php

class colours{

	public $name = "Green<br>";

	function name_code(){

		return $this->name;

	}

}

$object = new colours();

echo $object->name_code();

?>

<?php

$renk = "Yeşil";
```

```
function renk_kodu(){
	global $renk;
	return $renk;
}
echo renk_kodu();
?>
```

---

class2.php

```
<?php
class isimler{
	public $isim = "Semih";

	function ismicagir(){
		return $this->isim;
	}
	function yeni_isim($yeni){
		$this->isim=$yeni;
	}
}
$nesne = new isimler();
$nesne->isim="Semih";
$nesne->yeni_isim("Ahmet");
echo $nesne->ismicagir();
?>
```
