

WEB SAYFASI TASARIMI

DERS NOTU

WST01: PHP Diline Giriş

Öğr. Gör. Gökhan GÜVEN

PHP	1
PHP'yi Windows İşletim Sistemine Kurmak	1
PHP için Editör (Notepad++)	7
PHP Kodlama Kuralları	7
PHP içindeki Yorum Satırları	8
PHP'nin Büyük/Küçük Harf Duyarlılığı	8
Değişken Tanımlama	9
Değişken Çıktıları	9
Küresel ve Yerel Kapsam	10
Global Anahtar Kelimesi	11
Static Anahtar Kelimesi	11
echo ve print İfadeleri	12
Veri Türleri	13
Metin Fonksiyonları	15
Sabitler (Constants)	16
Operatörler	17
Aritmetik operatörler	17
Atama Operatörleri	17
Karşılaştırma operatörleri	18
Artırma Azaltma Operatörleri	18
Mantıksal Operatörler	18
Metin Operatörleri	19
Dizi Operatörleri	19
Koşullu İfadeler	19
If	20
If...else	20
If...elseif...else	21
Switch	21
Döngüler	23
while Döngüsü	23
do...while döngüsü	23
for Döngüsü	24

foreach Döngüsü	25
Fonksiyonlar	25
Kullanıcı Tanımlı Fonksiyonlar	25
Kullanıcı Tanımlı Fonksiyon Oluşturma	25
Fonksiyon Argümanları	26
Varsayılan Argüman Değeri	26
Fonksiyon İçinden Döndürülen Değerler	27
Diziler	27
Dizi Oluşturmak	28
İndeksli Diziler	28
Dizinin Uzunluğunu Almak – count() fonksiyonu	28
İndeksli Dizilerde Döngü	28
Çağrışımsal Diziler – (Associative Arrays)	29
Çağrışımsal Dizilerde Döngü	29
Dizilerde Sıralama	29
Dizilerde Artan Sıralama – sort()	30
Dizilerde Azalan Sıralama – rsort()	30
Çağrışımsal Dizilerin Değere Göre Artan Sıralaması	31
Çağrışımsal Dizilerin Anahtara Göre Artan Sıralaması	31
Çağrışımsal Dizilerin Değere Göre Azalan Sıralaması	31
Çağrışımsal Dizilerin Anahtara Göre Azalan Sıralaması	32
Küresel (Global) Değişkenler - Süper Küreseller (Superglobals)	32
\$GLOBALS	32
\$_SERVER	33
\$_REQUEST	34
\$_POST	35
\$_GET	35

PHP

Web sayfaları dinamik ve statik olmak üzere ikiye ayrılır. Statik sayfalar oluşturulurken doğrudan html yapısı kullanılır. Bu sayfaların değiştirilmesi için mutlaka web sitesini tasarlayan kişi ya da kişilerin doğrudan bir editör yazılımı ile müdahalesi şarttır.

Dinamik sayfalar sunucu (server) ismi verilen yazılım ve donanım tarafından, web sitesi kullanıcısının istekleri doğrultusunda html kodları üretebilen yapılardır. PHP (Hypertext Preprocessor), bu şekilde sayfalar oluşturmaya olanak tanıyan, dünyada yaygın olarak kullanılan, sunucu üzerinde çalışan ücretsiz bir betik (script) dilidir. Temeli Perl adı verilen bir programlama diline dayanır.

PHP konusunda çalışmaya başlamadan önce;

- HTML
- CSS
- JavaScript

alanlarında temel bilgiye sahip olmanız gerekir.

PHP dosyaları; metin, HTML, CSS, JavaScript ve PHP kodları içerebilir. PHP kodları sunucu üzerinde çalışır. Sonuçlar sıradan bir browser tarafından görüntülenebilen HTML kodları şeklinde döndürülür. PHP dosyalarının uzantısı mutlaka “.php” olmalıdır.

PHP dinamik sayfa içeriği üretebilir. Sunucu üzerinde dosya oluşturabilir, açabilir, okuyabilir, yazabilir, silebilir ve kapatabilir. Form verilerini toplayabilir. Çerez gönderebilir ve alabilir. Veritabanına veri ekleyebilir, silebilir ya da düzenleyebilir. Kullanıcı erişimli kontroller ve şifrelenmiş veri yapısını kullanabilir.

PHP'nin HTML çıktısı sınırsızdır. Çıktı olarak resim, PDF dosyaları, Flash animasyonları verebilir. XHTML ve XML'i destekler.

PHP çok farklı platformlarda çalışır. (Windows, Linux, Unix, Mac OS vs...) Birçok web sunucusula uyumludur. (Apache, IIS vs...) Neredeyse tüm veritabanlarını destekler. PHP ücretsizdir. Resmi PHP sitesinden rahatlıkla indirilebilir (www.php.net) Öğrenmesi ve sunucu tarafı çalıştırılması oldukça kolaydır.

PHP çalışmanız için ya PHP ve MySQL desteği olan bir hosting hizmeti almalısınız, ya da kendi bilgisayarınıza PHP ve MySQL desteği olan bir web sunucu kurmalısınız. Çalışmalarınızda rahat etmek için kendi bilgisayarınıza kurmanız daha verimli olacaktır.

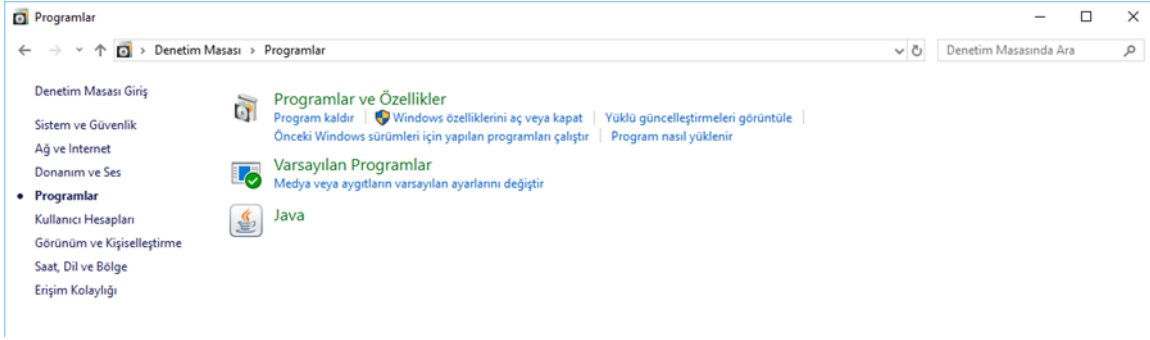
Bilgisayarınıza PHP destekli bir web sunucu hizmeti kurmak için <http://php.net/manual/tr/install.php> sayfasını ziyaret edebilirsiniz.

Bir PHP betiği sunucu üzerinde çalıştırılır. Sunucu çalıştırdığı kodlara ilişkin sonuçları HTML olarak tarayıcıya gönderir.

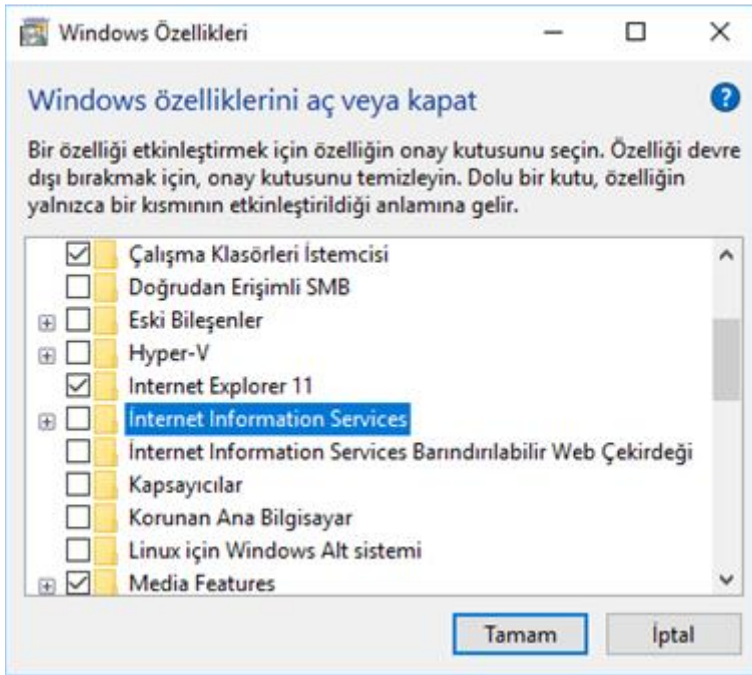
PHP'yi Windows İşletim Sistemine Kurmak

PHP Web sunucusu üzerinde çalışan bir script dilidir. Bu yüzden mutlaka bir web sunucusu hizmetine ihtiyacınız var. Burada Windows çalıştıran bir bilgisayara bu hizmetin nasıl kurulacağını daha sonra da PHP'nin web sunucusuna nasıl entegre edileceğini öğreneceksiniz.

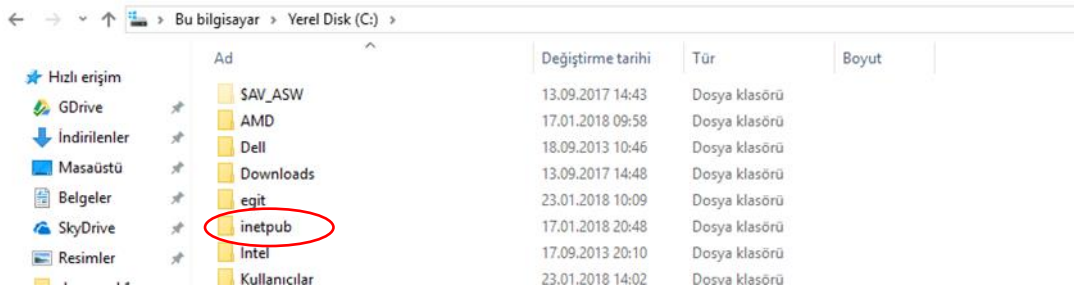
Windows işletim sistemlerinin kullandığı web sunucusu hizmetinin adı “IIS”dir. (Internet Information Services) Bu hizmeti kurmak için Denetim Masası→Programlar’a gidin ve Programlar ve Özellikler başlığı altındaki “Windows Özelliklerini aç veya Kapat” ı tıklayın.



Açılan pencereden Internet Information Services'i bulup önündeki kutucuğu işaretleyin ve daha sonra Tamam tuşuna basın.



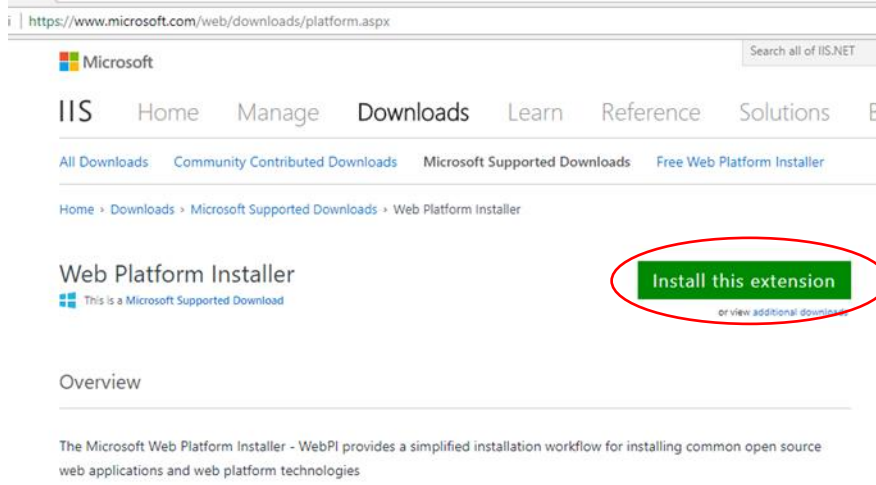
Bilgisayar Internet Information Services hizmetini yükleyecektir. Eğer sorunsuz bir yükleme gerçekleştirildiyse işletim sisteminin yüklü olduğu kök dizinde (genellikle C:) aşağıdaki gibi bir klasör belirmiş olmalıdır.



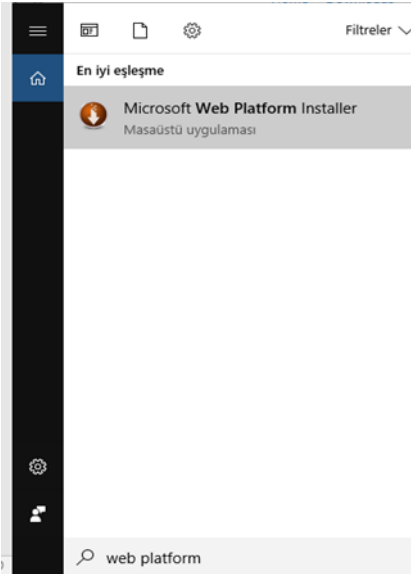
Yine de yüklemenin başarısını test etmek için herhangi bir tarayıcıyı açıp adres kısmına "localhost" ya da "127.0.0.1" yazıp Enter tuşuna basabilirsiniz. Size bilgisayarınızda yüklü olan IIS versiyonunu gösteren bir sayfayla karşılaşmalısınız. Eğer karşılaşmazsanız adres çubuğuna yazdığınız ifadeyi tekrar kontrol edin. Hala sorun varsa kurulumu kaldırıp tekrar yükleyebilirsiniz.

Kurulumunu gerçekleştirdiğiniz bu servis sayesinde bilgisayarınız web sayfalarını yayımlamaya hazır olacaktır. Ancak PHP sayfalarını PHP desteği vermeden çalıştıramazsınız. PHP'nin kurulumunu en kolay

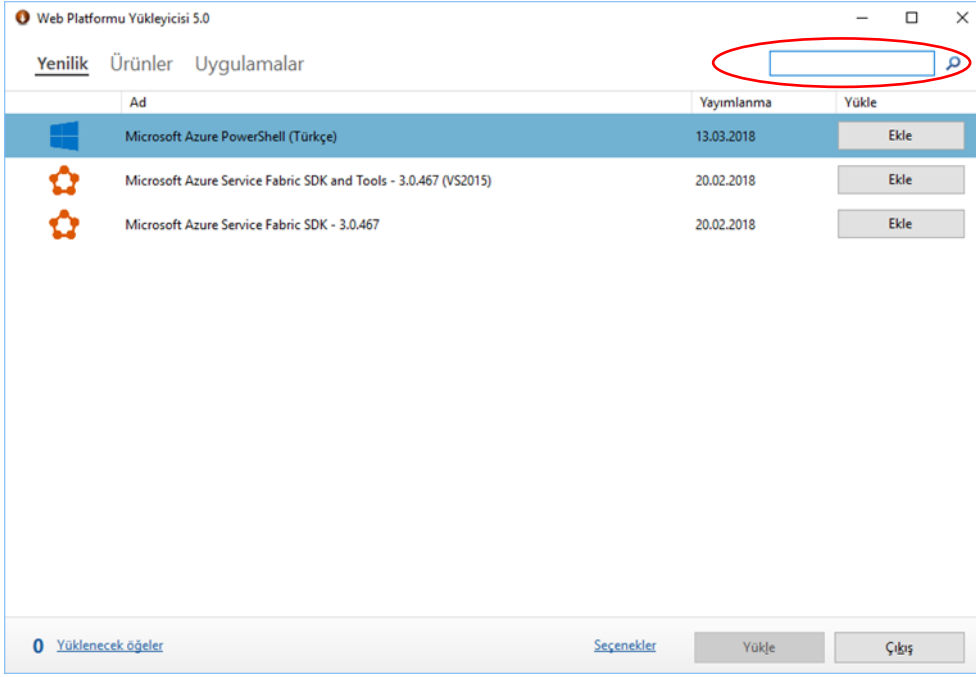
yine Microsoft firmasının bir yardımcı programı olan “Web Platform Installer” yazılımı ile yapabilirsiniz. Bunun için internette “Web Platform Installer” şeklinde bir arama yapın ve Microsoft’un ilgili sayfasına gidin. Aşağıdakine benzer bir sayfayla karşılaşacaksınız. Bu sayfadan Install this extension düğmesini tıklayın.



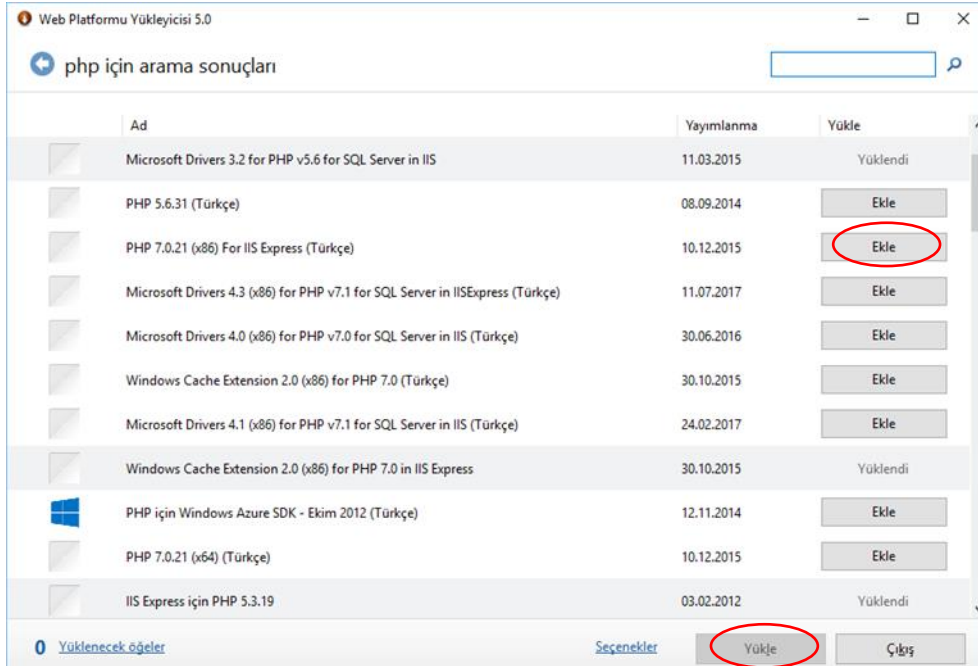
İsminde “WebPlatformInstaller” ifadesi geçen bir dosyayı bilgisayarınıza indireceksiniz. Bu dosyayı çalıştırın. Gerekli direktifleri izleyerek kurulumu tamamlayın. İsmi aşağıdaki gibi olan programı çalıştırın.



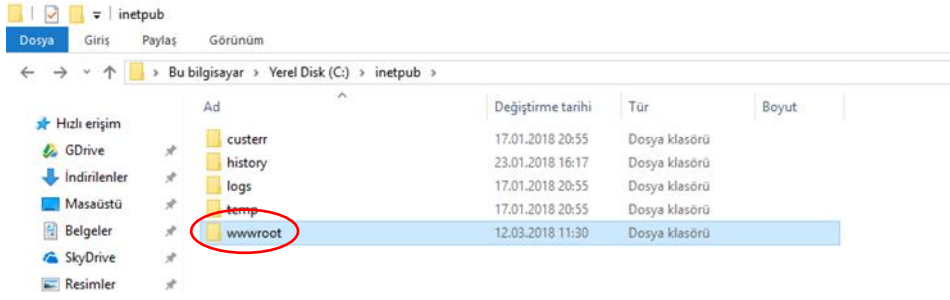
Aşağıdaki gibi bir pencereyle karşılaşacaksınız. Arama kutucuğuna php yazıp Enter’a basın.



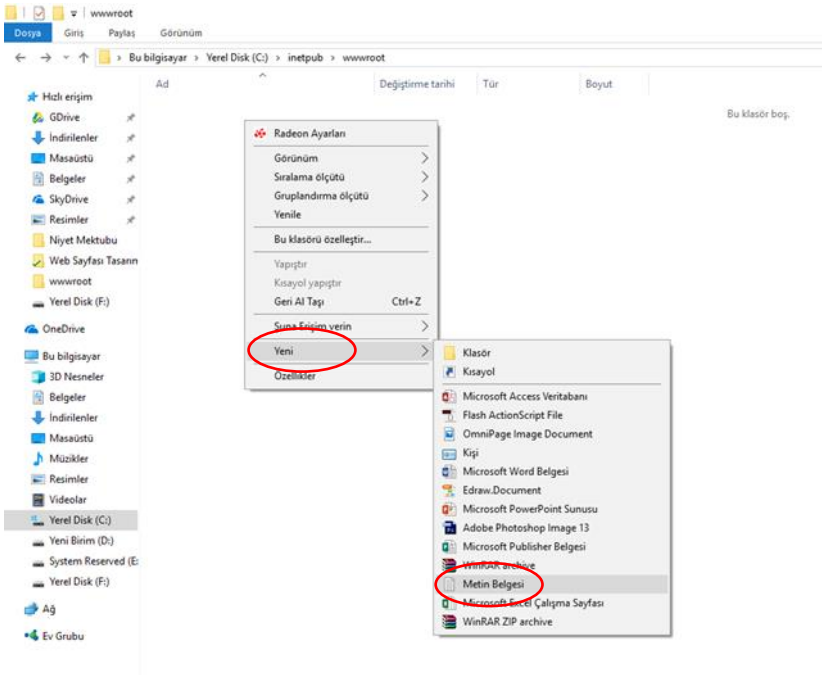
Php ile ilgili birçok arama sonucu döndürülecektir. Sonuçlardan aşağıdakine benzer ve IIS için olan PHP versiyonlarından herhangi birisinin yanındaki Ekle düğmesine ve daha sonra Yükle düğmesine tıklayın.



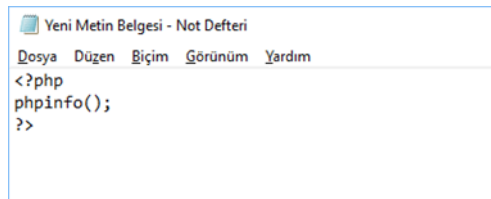
Bilgisayar gerekli indirme ve yükleme işlemlerini gerçekleştirecektir. Yüklemenizin doğru çalışıp çalışmadığını anlamak için sırasıyla C:→inetpub→wwwroot klasörüne gidin. Klasörün içine girin.



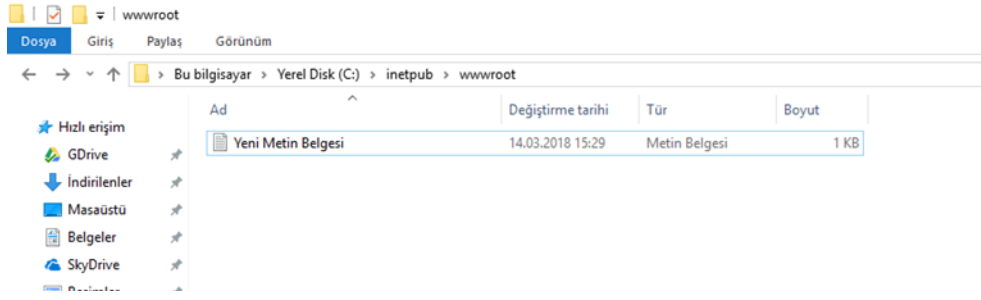
Bu klasörün içindekileri seçip silin. Yayımlamak istediğiniz web sayfalarını bu klasör içinde barındıracaksınız. Klasörün içine fareyle sağ tuş yapıp sırasıyla



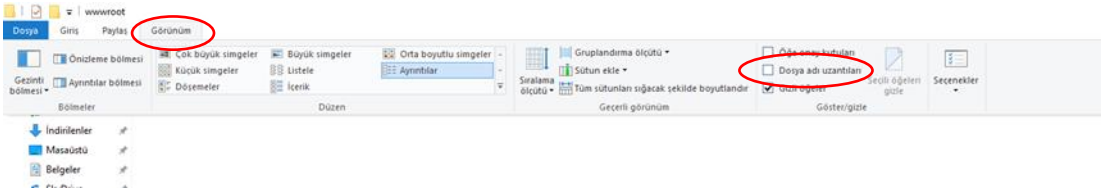
“Yeni Metin Belgesi” şeklinde bir dosya oluşturmuş olacaksınız. Şimdi bu dosyanın içine girin ve aşağıdaki gibi kodları girin, kaydedip çıkın.



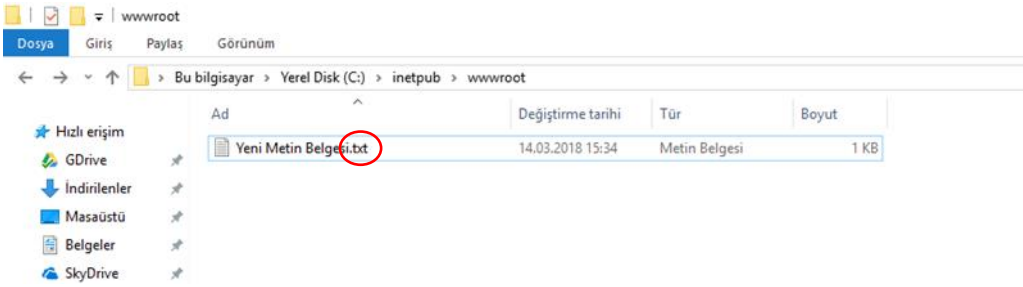
Şimdi dosyanın ismini değiştirmeniz gerekiyor. Uzantının mutlaka “.php” olduğundan emin olmalısınız. Muhtemelen “Yeni Metin Belgesi” isimli dosyayı şu şekilde görüyorsunuz.



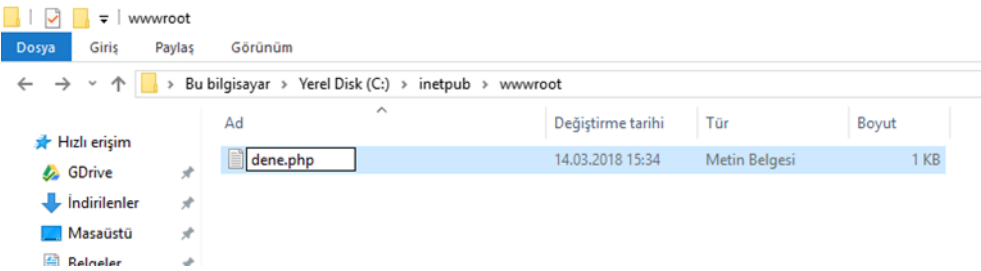
Dosyaların uzantısını görmek çok önemlidir. Bunun için basit bir ayar yapmamız gerekiyor. Üst taraftaki sekmelerden Görünüm'ü tıklayın ve Dosya Adı Uzantıları ifadesinin önündeki kutucuğu işaretleyin.



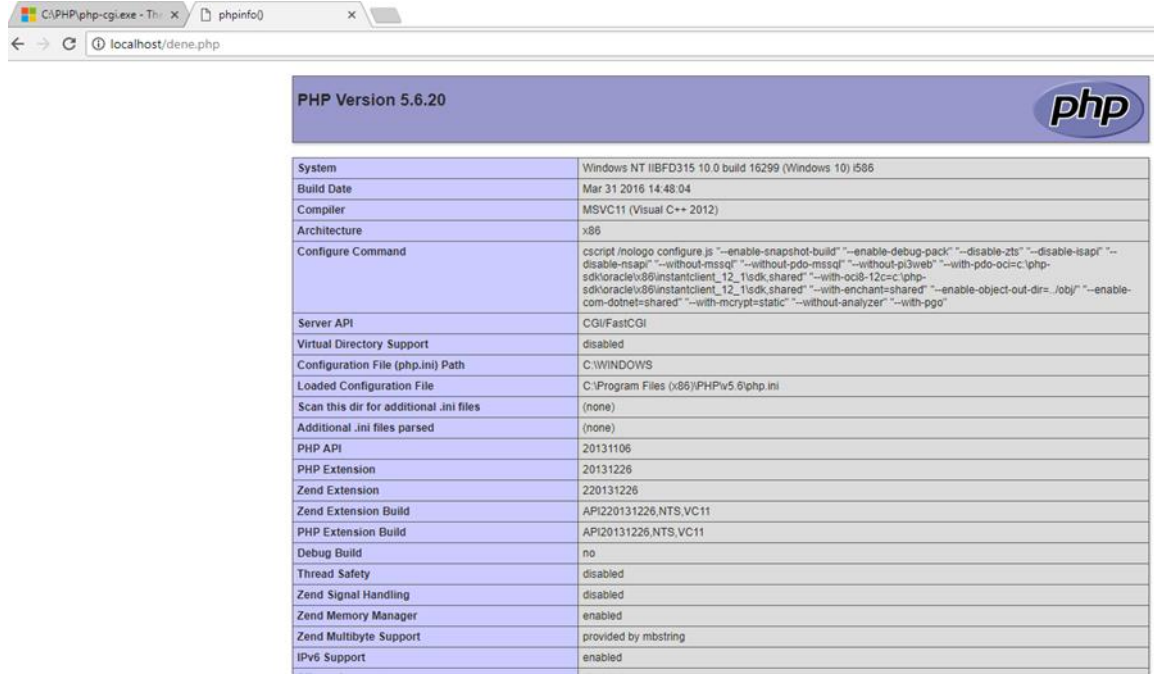
Bunu yapar yapmaz uzantıları görebilir hale geldiniz. Şimdi "Yeni Metin Belgesi" dosyasına dikkat edin uzantısının "txt" olduğunu göreceksiniz.



Dosyanın üzerine fareyle sağ tıklayıp "Yeniden Adlandır" komutunu verin.



Dosyayı bu şekilde adlandırdıktan sonra herhangi bir tarayıcı penceresi açın ve şu adresi "<http://localhost/dene.php>" adres çubuğuna girip Enter tuşuna basın. Aşağıdaki gibi bir sayfayla karşılaşıyorsanız artık bilgisayarınızda php çalıştırabilirsiniz demektir.



PHP Version 5.6.20	
System	Windows NT IIBFD315 10.0 build 16299 (Windows 10) i586
Build Date	Mar 31 2016 14:48:04
Compiler	MSVC11 (Visual C++ 2012)
Architecture	x86
Configure Command	cmd.exe /c "cd /d %~dp0 & phpize & ./configure --enable-snapshot-build --enable-debug-pack --disable-zts --disable-igmp --disable-nsapi --without-mssql --without-pdo-mssql --without-p3web --with-pdo-oci=c:\php-sdk\oracle\99\instantclient_12_1\sdk\shared --with-oci8-12c=c:\php-sdk\oracle\99\instantclient_12_1\sdk\shared --with-enchant=shared --enable-object-out-dir=.obj --enable-com-dotnet=shared --with-mcrypt=static --without-analyzer --with-pgsql"
Server API	CGI/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	C:\WINDOWS
Loaded Configuration File	C:\Program Files (x86)\PHP\5.6\php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20131106
PHP Extension	20131226
Zend Extension	220131226
Zend Extension Build	API220131226,NTS,VC11
PHP Extension Build	API20131226,NTS,VC11
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled

PHP için Editör (Notepad++)

PHP çalışırken kolaylık olması açısından bir editöre başvurmak iyi olacaktır. Bunun için “Notepad++” isimli programı kullanabilirsiniz. Bu program yazdığınız kodları daha net görmeyi sağlayacaktır. Bu program ücretsiz ve küçük bir programdır. İndirmek için <https://notepad-plus-plus.org/download/> adresini ziyaret edebilirsiniz.

PHP Kodlama Kuralları

Bir PHP betiği belgenin herhangi bir yerine yazılabilir. Mutlaka “<?php” ile başlamalı ve “?>” ile bitmelidir.

ÖRNEK

```
<?php
// PHP Kodları buraya yazılır
?>
```

Varsayılan dosya uzantısı “.php” dir.

Sıradan bir PHP dosyası HTML etiketleri ve PHP betik kodları içerir. Aşağıda sıradan bir PHP dosyası görmekteyiz. “echo” komutu “Merhaba Dünya” yazısının ekrana yazılmasını sağlayan PHP komutudur. PHP komutlarının en sonunda noktalı virgül “;” olmak zorundadır.

ÖRNEK

```
<html>
<body>
<h1>Bu benim ilk php kodum</h1>
<?php
echo "Merhaba Dünya!";
?>
</body>
</html>
```

PHP içindeki Yorum Satırları

Programlama yapılırken yorum satırları kullanmak programlamanın anlaşılması ve hatırlanması için önemlidir. Yorum satırları sunucu tarafından görmezden gelinen satırlardır. PHP'de farklı yöntemlerle yorum satırı oluşturulabilir.

ÖRNEK

```
<html>
<body>
<?php
// Tek satırdan oluşan yorum satırı

# Diyez işareti de tek satırdan oluşan yorum satırı için kullanılabilir.

/*
Burada birden fazla
yorum satırı yazmak için
bir yöntem görüyorsunuz.
*/

// Yorumunuzu doğrudan kodun içine de gömebilirsiniz.
$x = 5 /* + 15 */ + 5;
echo $x;
?>
</body>
</html>
```

PHP'nin Büyük/Küçük Harf Duyarlılığı

PHP'nin komutları, sınıfları, fonksiyonları ya da kullanıcı tanımlı fonksiyonları büyük/küçük harf duyarlı değildir. Yani ister büyük harf ya da küçük harfle yazabilirsiniz.

ÖRNEK

```
<html>
<body>
<?php
    ECHO "Merhaba Dünya!<br>";
    echo "Merhaba Dünya!<br>";
    EcHo "Merhaba Dünya!<br>";
?>
</body>
</html>
```

Buna karşın değişken isimleri büyük/küçük harf duyarlıdır. Yani değişkeni ne şekilde tanımlamışsanız o şekilde çağırmanız gerekir.

ÖRNEK

```
<html>
<body>
<?php
    $renk = "siyah";
    echo "Arabam " . $renk . "<br>";
    echo "Arabam " . $RENK . "<br>";
    echo "Tarzım " . $reNK . "<br>";
?>
</body>
</html>
```

Dikkat ederseniz değişken çağrılırken \$renk, \$RENK, \$reNK kullanılmıştır. Sadece ilk satırdaki renk yazdırılmıştır. Çünkü bilgisayar tarafından \$renk, \$RENK, \$reNK üç farklı değişken olarak değerlendirilmiştir.

Değişken Tanımlama

PHP'de değişkenler mutlaka \$ (dolar) işaretiyle başlar.

ÖRNEK

```
<?php
    $metin = "Merhaba dünya!";
    $x = 7;
    $y = 9.33;
?>
```

Yukarıdaki örnek kodlarda \$metin değişkeninin değeri "Merhaba dünya!", \$x değişkeninin değeri "7 sayısı", \$y değişkeninin değeri "9,33 sayısı"dır.

Diğer programlama dillerinin aksine, PHP'de program içinde kullanılacak değişkenler önceden bildirilmez. Değişkene değer atandığı anda değişken oluşturulmuş olur.

Bir değişkene metin değeri atarken mutlaka atanmış metnin tırnak içinde yazılması gerekir.

Değişken tanımlama kurallarını kısaca toparlayacak olursak;

- Değişken \$(dolar) işaretiyle başlamalıdır.
- Değişken ismi mutlaka bir harf ya da alt çizgi (_) ile başlamalıdır.
- Değişken isminde sayılar geçebilir, ancak sayıyla başlayamaz.
- Türkçe ve özel karakterler içeremez. (Ç, ç, İ, Ö, ö, Ü, ü, Ğ, ğ, \$, ş, !, +, - vs...)
- Büyük/küçük harf duyarlıdır. (\$hesap ve \$HESAP farklı değişkenlerdir.)

Değişken Çıktıları

PHP'de echo ifadesi ekranda çıktı görmek için en fazla kullanılan ifadedir.

Aşağıda bir değişkenin değerinin metin içinde nasıl gösterileceğini görüyorsunuz.

ÖRNEK

```
<?php
    $metin = "İİBF";
    echo "Benim fakültem $metin!";
?>
```

Yukarıdaki kodun çıktısının aynısını veren aşağıdaki kodu inceleyin.

ÖRNEK

```
<?php
    $metin = "İİBF";
    echo "Benim fakültem $metin!";
?>
```

İki değişkenin toplam değerini aşağıdaki kodları yazarak ekranda gösterebiliriz.

ÖRNEK

```
<?php
    $x = 3;
    $y = 12;
    echo $x + $y;
?>
```

PHP esnek bir dildir. Değişken türleri atama işlemi esnasında otomatik olarak belirlenir. C, C++ ve Java gibi diğer programlama dillerinde değişkenin türü ve ismi, değişken kullanılmadan önce mutlaka bildirilir.

PHP değişkenler kodlar içinde herhangi bir yerde tanımlanabilir. Bir değişkenin kapsamı betik içinde nasıl kullanılacağıyla ilgilidir ve üç türü vardır.

- Yerel (local)
- Küresel (global)
- Statik (statik)

Küresel ve Yerel Kapsam

Global (Küresel) değişken bir fonksiyon dışında tanımlanmışsa sadece fonksiyon dışında erişilebilir haldedir. Yani yalnızca fonksiyonun dışında kullanılır.

ÖRNEK

```
<?php
    $x = 5; // küresel kapsam
    function deneme () {
        // fonksiyon içinde kullanılan $x değişkeni hata döndürecek
        echo "<p>x değişkeni fonksiyon içinde: $x</p>";
    }
    deneme ();
    echo "<p>x değişkeni fonksiyon dışında: $x</p>";
?>
```

Bir değişken bir fonksiyon içinde tanımlanmışsa yerel kapsamlıdır. Yani yalnızca fonksiyon içinde kullanılabilir.

ÖRNEK

```
<?php
function deneme () {
    $x = 5; // yerel kapsam
    echo "<p>x değişkeni fonksiyon içinde: $x</p>";
}
deneme ();
// x değişkeninin fonksiyon dışında kullanılması hata döndürecektir
echo "<p>x değişkeni fonksiyon dışında : $x</p>";
?>
```

Global Anahtar Kelimesi

Bir global değişkeni fonksiyon içinde çağırmak için "global" anahtar kelimesi kullanılır.

ÖRNEK

```
<?php
$x = 5;
$y = 10;
function deneme () {
    global $x, $y;
    $y = $x + $y;
}
deneme ();
echo $y; // çıktı 15
?>
```

Global değişkenler \$GLOBALS[index] dizisinde saklanır. "index" ifadesi değişkenin ismidir. Bu diziye fonksiyon içinden erişilebilir ya da global değişkenin doğrudan düzenlenmesi için kullanılabilir.

ÖRNEK

```
<?php
$x = 5;
$y = 10;
function deneme () {
    $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];
}
deneme ();
echo $y; // çıktı 15
?>
```

Static Anahtar Kelimesi

Bir fonksiyon çalıştırıldığında ve sonuçlandığında, fonksiyon içindeki değişkenlerin tamamı silinir. Ancak bazen yerel değişkenin değerinin silinmesini istemeyiz. Bu durumda static anahtar kelimesini kullanırız.

ÖRNEK

```
<?php
function deneme () {
    static $x = 0;
    echo $x;
    $x++; //değişkenin değerini 1 artırmak anlamına gelir.
}
deneme ();
deneme ();
deneme ();
?>
```

Yukarıdaki örnekte fonksiyon her çağırıldığımızda kullanılan yerel değişkenin değerinin saklandığına dikkat edin.

echo ve print ifadeleri

PHP'de çıktı almak iki temel yolu echo ve print ifadelerini kullanmaktır. echo ifadesi en yaygın kullanılanıdır. Çünkü çok fazla ve farklı parametreyle kullanımı söz konusudur. Aynı zamanda print ifadesine göre oldukça hızlıdır.

echo parantezle ya da parantez olmadan kullanılabilir. Yani echo ve echo() aynı sonucu verir.

ÖRNEK

```
<?php
echo "<h2>Selam!</h2>";
echo "Merhaba Dünya!<br>";
echo "Bu benim ilk PHP kodum!<br>";
echo "Buradaki ", "metin ", "birden ", "çok ", "parametreyle
oluşturuldu.";
?>
```

Yukarıdaki örnekte tırnak içinde HTML kodlarının yazıldığına dikkat edin.

ÖRNEK

```
<?php
$metin1 = "PHP öğren";
$metin2 = "hergün";
$x = 5;
$y = 4;
echo "<h2>" . $metin . "</h2>";
echo "PHP çalış " . $metin2 . "<br>";
echo $x + $y;
?>
```

Print ifadesi parantezle veya parantez olmadan kullanılabilir. Yani print ve print() aynı şeylerdir.

ÖRNEK

```
<?php
print "<h2>Selam!</h2>";
print "Merhaba Dünya!<br>";
print "Bu benim ilk PHP kodum!<br>";
?>
```

echo ile kullanımı aynıdır.

ÖRNEK

```
<?php
$metin1 = "PHP öğren";
$metin2 = "hergün";
$x = 5;
$y = 4;

print "<h2>" . $metin . "</h2>";
print "PHP çalış " . $metin2 . "<br>";
print $x + $y;
?>
```

Veri Türleri

Değişkenler farklı türde veriler saklayabilir. PHP'nin desteklediği veri tipleri şunlardır;

- Metin (String)
- Tamsayı (Integer)
- Ondalık Sayı (Float ya da Double)
- Mantıksal (Boolean)
- Dizi (Array)
- Nesne (Object)
- Boş (NULL)
- Kaynak (Resource)

Metin veri tipi bir karakter dizisidir. "Merhaba Dünya!" metindir (string). Metinler mutlaka tırnaklar içine yazılır. PHP'de kesme (') ya da (") aynı görevi yerine getirir.

ÖRNEK

```
<?php
$x = "Merhaba dünya!";
$y = 'Merhaba dünya!';
echo $x;
echo "<br>";
echo $y;
?>
```

Tamsayı veri türü ondalık kısım içermeyen -2,147,483,648 ve 2,147,483,647 bir sayıdır. En az bir basamağı olmak zorundadır. Ondalık kısmı olmamalıdır. Pozitif ya da negatif bir sayı olabilir. Tam sayılar üç farklı biçimde olabilir;

- Decimal (Onluk sayı sistemi)

- Hexadecimal (Onaltılık sayı sistemi)
- Octal (Sekizlik sayı sistemi)

PHP’de bir değişkenin veri türünü ve değerini görmek için `var_dump()` fonksiyonu kullanılır. Aşağıdaki örnekte `$x` değişkeni bir tam sayıdır.

ÖRNEK

```
<?php
    $x = 5985;
    var_dump($x);
?>
```

Float değişken ondalık sayı içerir. Üstsel biçimde olabilir. Aşağıdaki kodları çalıştırın ve döndürülen sonuca dikkat edin.

ÖRNEK

```
<?php
    $x = 10.365;
    var_dump($x);
?>
```

Boolean iki durum içerebilir; Doğru (true) ya da Yanlış (false). Bunu 1 ya da 0 gibi de düşünebilirsiniz. En fazla mantıksal sınamalar için kullanılırlar.

```
$x = true;
```

```
$y = false;
```

Diziler çok sayıda farklı veriyi tek değişken ismiyle saklarlar.

Aşağıdaki örnekte `$araba` değişkeni bir dizidir. Örnekteki `var_dump()` fonksiyonu veri türünü ve değerini döndürür.

ÖRNEK

```
<?php
    $araba = array("Skoda", "Mercedes", "Fiat");
    var_dump($araba);
?>
```

Nesne (Object) türü verinin nasıl işlendiği ile ilgili bilgiyi saklamak için kullanılır. Eğer nesne türü kullanılacaksa mutlaka açıkça deklare edilmelidir. Nesnenin deklare edilmesi class kullanılarak yapılır. Bunun için ilk olarak class (sınıf) anahtarını kullanırız. Sınıf (class) özellikleri ve yöntemleri içeren bir yapıdır.

ÖRNEK

```
<?php
class Araba {
    function Araba () {
        $this->model = "VW";
    }
}

// bir nesne oluşturuluyor
$golf = new Araba ();

// nesne özellikleri gösteriliyor
echo $golf->model;
?>
```

Boş (NULL) değer, özel bir veri türüdür. Bir değişkenin hiç birşey içermemesini sağlamak için bu veri tipi atanır. Eğer bir değişken oluşturulurken herhangi bir değer atanmamışsa değeri "NULL" yani boştur. Aşağıda bir değişkenin değerinin boş olmasının sağlanmasını görüyorsunuz.

ÖRNEK

```
<?php
$x = "Merhaba dünya!";
$x = null;
var_dump($x);
?>
```

Kaynak (Resource) veri tipi, veritabanı istekleri için kullanılır.

Metin Fonksiyonları

Bir metnin kaç karakterden oluştuğunu görmek için `strlen()` fonksiyonunu kullanın.

ÖRNEK

```
<?php
echo strlen("Merhaba dünya!"); // çıktısı 14
?>
```

Bir metnin kaç kelimedenden oluştuğunu görmek için `str_word_count()` fonksiyonunu kullanın.

ÖRNEK

```
<?php
echo str_word_count("Merhaba dünya!"); // çıktısı 2
?>
```

Bir metnin tam tersini elde etmek için `strrev()` fonksiyonunu kullanın.

ÖRNEK

```
<?php
echo strrev("Selam"); // çıktısı maleS
?>
```

Metnin içinde özel bir metin aramak için strpos() fonksiyonunu kullanın. İlk karakterin pozisyonunun 1'den değil 0'dan başlayarak sayıldığına dikkat edin.

ÖRNEK

```
<?php
echo strpos("Merhaba dünya!", "dünya"); // çıktısı 8
?>
```

Metin içindeki bir ifadenin yerine başkasını yerleştirmek için str_replace() fonksiyonunu kullanın.

ÖRNEK

```
<?php
echo str_replace("dünya", "Çocuk", "Merhaba dünya!"); // çıktısı
Merhaba Çocuk!
?>
```

PHP'de daha çok sayıda metin fonksiyonu bulunur. Bunlar için Internet'te küçük bir araştırma yapabilirsiniz.

Sabitler (Constants)

Sabitler betik boyunca değiştirilmesi istenmeyen bir değer atandığı yapılardır. Değişken \$ işaretiyle başlarken, sabit isminde bu işaret kullanılmaz. Değişkenlerin aksine sabitler, otomatik olarak betik içinde küresel kapsamlıdır.

Sabit tanımlamak için define() fonksiyonu kullanılır.

SÖZ DİZİMİ

```
define(isim, değer, büyük/küçük harf duyarlılığı)
```

Parametreler

- İsim: Sabiti ifade eden bir isim
- Değer: Sabite atanacak değer
- Büyük/küçük harf duyarlılığı: Sabitin adı istenirse büyük/küçük harf duyarlılığı olmadan kullanılabilir. Varsayılan değeri false yani büyük küçük harf duyarlıdır.

ÖRNEK

```
<?php
define("MERHABA", "İİBF'ye hoş geldiniz!");
echo MERHABA;
?>
```

Yukarıdaki örnekte sabit büyük/küçük harf duyarlıdır. Aşağıda duyarlı olmayan tanımlamayı görüyorsunuz.

ÖRNEK

```
<?php
define("MERHABA", "İİBF'ye hoş geldiniz!", true);
echo merhaba;
?>
```

Sabitler kendiliğinden küreseldir. Betik boyunca istediğiniz yerde çağırabilirsiniz. Herhangi bir fonksiyonun içinden çağırabilirsiniz. Burada dikkat etmeniz gereken şey sabit tanımının fonksiyon dışında bir yerlerde yapılması gerektiğidir.

ÖRNEK

```
<?php
define("MERHABA", "İİBF'ye hoş geldiniz!");
function deneme () {
    echo MERHABA;
}
deneme ();
?>
```

Operatörler

Operatörler değişkenler ve değerler arasında ilişki kurmak ve çeşitli işlemler yapmak için kullanılır. Operatör grupları şu şekilde sıralanabilir;

- Aritmetik operatörler
- Atama operatörleri
- Karşılaştırma operatörleri
- Artırma/Azaltma operatörleri
- Mantıksal operatörler
- Metin operatörleri
- Dizi operatörleri

Aritmetik operatörler

Operatör	Eylem	Örnek	Sonuç
+	Toplama	$\$x + \y	İki değişkenin değerini toplar
-	Çıkarma	$\$x - \y	Birinci değişkenden ikincisini çıkarır
*	Çarpma	$\$x * \y	Değişkenlerin değerlerini çarpar
/	Bölme	$\$x / \y	Birinci değişkeni ikinci değişkene böler
%	Mod	$\$x \% \y	Birinci değişkenin ikinci değişkene bölümünden kalanı verir
**	Üs	$\$x ** \y	Birinci değişkenin ikinci değişkene göre kuvvetini verir

Atama Operatörleri

PHP'de temel atama operatörü "=" simgesidir. Sağdaki değer soldaki değişkene atanır.

Atama Operatörü	Operatör Kullanılmadan Yapılabilen	Açıklama
$\$x = \y	$\$x = \y	$\$x$ 'in değeri $\$y$ 'dir.
$\$x += \y	$\$x = \$x + \$y$	$\$x$ 'in yeni değeri kendi değeri ve $\$y$ 'nin toplamıdır.
$\$x -= \y	$\$x = \$x - \$y$	$\$x$ 'in yeni değeri kendi değerinden $\$y$ 'nin eksiltilmiş halidir.

$\$x *= \y	$\$x = \$x * \$y$	$\$x$ 'in yeni değeri kendi değeri ile $\$y$ 'nin çarpımıdır.
$\$x /= \y	$\$x = \$x / \$y$	$\$x$ 'in yeni değeri kendi değerinin $\$y$ 'ye bölümüdür.
$\$x \% = \y	$\$x = \$x \% \$y$	$\$x$ 'in yeni değeri kendi değerinin $\$y$ 'ye bölümünden kalandır.

Karşılaştırma operatörleri

Metin ya da sayı değerlerini birbirleriyle karşılaştırmak için kullanılır.

Operatör	Karşılaştırma	Kullanımı	Sonuç
==	Eşit	$\$x == \y	$\$x$ 'in değeri $\$y$ 'ye eşitse doğru döndürür.
===	Denk	$\$x === \y	$\$x$ ve $\$y$ aynı tür değer taşıyorsa ve birbirlerine eşitse doğru döndürür.
!=	Eşit değil	$\$x != \y	$\$x$ 'in değeri $\$y$ 'ye eşit değilse doğru döndürür.
<>	Eşit değil	$\$x <> \y	$\$x$ 'in değeri $\$y$ 'ye eşit değilse doğru döndürür.
!==	Denk Değil	$\$x !== \y	$\$x$ ve $\$y$ aynı tür değer taşıyorsa ya da birbirlerine eşit değilse doğru döndürür.
>	Büyük	$\$x > \y	$\$x$ 'in değeri $\$y$ 'den büyükse doğru döndürür.
<	Küçük	$\$x < \y	$\$x$ 'in değeri $\$y$ 'den küçükse doğru döndürür.
>=	Büyük Eşit	$\$x >= \y	$\$x$ 'in değeri $\$y$ 'nin değerine eşitse ya da $\$y$ 'nin değerinden büyükse doğru döndürür.
<=	Küçük Eşit	$\$x <= \y	$\$x$ 'in değeri $\$y$ 'nin değerine eşitse ya da $\$y$ 'nin değerinden küçükse doğru döndürür.

Artırma Azaltma Operatörleri

Bir değişkenin değerini artırmak ya da azaltmak için kullanılan operatörlerdir.

Operatör	İşlem	Açıklama
++ $\$x$	Önceden Artırma	$\$x$ 'in değeri $\$x$ değişkeni kullanılmadan bir artırılır.
$\$x$ ++	Sonradan Artırma	$\$x$ 'in değeri $\$x$ değişkeni kullanıldıktan sonra bir artırılır.
-- $\$x$	Önceden Azaltma	$\$x$ 'in değeri $\$x$ değişkeni kullanılmadan bir azaltılır.
$\$x$ --	Sonradan Azaltma	$\$x$ 'in değeri $\$x$ değişkeni kullanıldıktan sonra bir azaltılır.

ÖRNEK

```
<?php
    echo "Önceden Artırma<br>";
    $x = 10;
    echo ++$x; //Çıktı 11

    echo "<br>"; //Satır Başı
    echo "<br>"; //Satır Başı

    echo "Sonradan Artırma<br>";
    $x=10;
    echo $x++; //Çıktı 10 ama bundan sonra $x'in değeri 11 oldu.
    echo "<br>"; //Satır başı
    echo $x; // Çıktı 11
?>
```

Mantıksal Operatörler

Mantıksal operatörler koşullu ifadeler ile birlikte kullanılır.

Operatör	Anlam	Kullanımı	Sonuç
and	ve	\$x and \$y	\$x ve \$y'nin değerlerinin her ikisi de doğruysa doğru döndürür.
or	ya da	\$x or \$y	\$x veya \$y'nin herhangi biri doğruysa doğru döndürür.
xor	Xor	\$x or \$y	\$x veya \$y'nin herhangi biri doğruysa doğru döndürür, her biri doğruysa yanlış döndürür.
&&	ve	\$x&&\$y	\$x ve \$y'nin değerlerinin her ikisi de doğruysa doğru döndürür.
	ya da (AltGr+Büyüktür Tuş Kombinasyonu)	\$x \$y	\$x veya \$y'nin herhangi biri doğruysa doğru döndürür.
!	değil	!\$x	\$x doğru değilse doğru döndürür.

Metin Operatörleri

Metinler üzerinde özel işlemler yapmak için kullanılan iki metin operatörü vardır.

Operatör	İşlem	Kullanımı	Sonuç
.	Birbirine bağlama	\$metin1 . \$metin2	\$metin1 değişkeni içindeki metni \$metin2 içindeki metin ile birleştirir.
.=	Birbirine bağlayarak atama	\$metin1 .= \$metin2	\$metin1 değişkeninin yeni değeri \$metin1 ve \$metin2 değişkeninin değerlerinin bağlanmış halidir.

Dizi Operatörleri

Dizileri karşılaştırmak ve birlikte kullanmak için kullanılan operatörlerdir.

Operator	Anlamı	Kullanımı	Sonuç
+	Bileşim	\$x+\$y	\$x ve \$y dizi değişkenlerinin bileşimini verir.
==	Eşitlik	\$x==\$y	Dizi değişkenlerinin anahtar/değer çiftleri birbirlerinin aynıysa doğru döndürür.
===	Denklik	\$x===\$y	Dizi değişkenler değer ve tür açısından birbirinin tıpkıysa doğru döndürür.
!=	Eşit Değil	\$x != \$y	\$x dizi değişkeninin değerleri \$y ile aynı değilse doğru döndürür.
<>	Eşit Değil	\$x <> \$y	\$x dizi değişkeninin değerleri \$y ile aynı değilse doğru döndürür.
!==	Denk Değil	\$x !== \$y	\$x dizisi \$y dizisi ile birebir aynı değilse doğru döndürür.

Koşullu İfadeler

Programlamada koşullu ifadeler, durum karşılaştırması yapıldıktan sonra her durumda neler yapılacağını belirtmek için kullanılır. Programlama için vazgeçilmezdir. Programlamanın temelini oluşturur.

PHP içinde kullanılan koşullu ifadeler şunlardır;

- "if" ifadesi: koşul doğruysa yapılacakları belirtmek için kullanılır.
- "if...else" ifadesi: koşul karşılanıyorsa yapılacakları, karşılanmıyorsa yapılacakları belirtmek için kullanılır.

- “if...elseif...else” ifadesi: birden çok farklı koşul durumunda yapılacakları belirtmek için kullanılır.
- “switch” ifadesi: birçok farklı kod grubunun hangi durumda gerçekleştirileceğini belirtmek için kullanılır.

If

If koşul sağlandığında yapılacakları belirtmek için kullanılan koşullu ifadedir.

SÖZ DİZİMİ

```
If(koşul) {  
    Koşul sağlandığında yapılacaklar (Koşul doğruysa yapılacaklar);  
}
```

Aşağıdaki örnekte serverdaki saat 20’den küçükse “İyi günler” yazan bir kod bulunuyor.

ÖRNEK

```
<?php  
    $s = date("H"); //Sunucudan saat bilgisi $s değişkenine atanıyor.  
  
    if ($s < "20") {  
        echo "İyi günler!";  
    }  
?>
```

If...else

Bu ifade koşul sağlandığında ve sağlanmadığında yapılacakları belirtmek için kullanılır.

SÖZ DİZİMİ

```
if (koşul){  
    Koşul karşılanıyorsa yapılacaklar;  
} else {  
    Koşul karşılanmıyorsa yapılacaklar;  
}
```

Aşağıdaki örnekte sunucunun saati 20 küçük ise “İyi günler!” değilse “İyi geceler!” yazdırılıyor.

ÖRNEK

```
<?php
    $s = date("H"); //Sunucudan saat bilgisi $s değişkenine atanıyor.

    if ($s < "20") {
        echo "İyi günler!";
    } else {
        echo "İyi geceler!";
    }
?>
```

If...elseif...else

Birden çok koşula göre kodlar çalıştırılacaksa bu ifade kullanılır.

SÖZ DİZİMİ

```
if (koşul) {
    Koşul doğruysa yapılacaklar;
} elseif (koşul) {
    Önceki koşul doğru değil ama bu doğruysa yapılacaklar;
} else {
    Daha önce yazılan koşulların hiçbiri sağlanmadıysa yapılacaklar;
}
```

Aşağıdaki örnekte saat bilgisi 10'dan küçükse "Günaydın" yok değil de 20'den küçükse "İyi Günler" bu da değilse "İyi geceler" yazdırılıyor.

ÖRNEK

```
<?php
    $s = date("H"); //Sunucudan saat bilgisi $s değişkenine atanıyor.

    if ($s < "10") {
        echo "Günaydın!";
    } elseif ($s < "20") {
        echo "İyi günler!";
    } else {
        echo "İyi geceler!";
    }
?>
```

Switch

Birbirinden farklı kod bloklarından birini seçip çalıştırmak için kullanılır.

SÖZ DİZİMİ

```
switch (n) {  
    case etiket1:  
        n=etiket1'se buradaki kodlar çalıştırılır.  
        break;  
    case etiket2:  
        n=etiket2'yse buradaki kodlar çalıştırılır.  
        break;  
    case etiket3:  
        n=etiket3'se buradaki kodlar çalıştırılır.  
        break;  
    ...  
    default:  
        n ile eşleşen herhangi bir durum yoksa buradaki kodlar çalıştırılır;  
}  
}
```

İlk önce söz dizimindeki “n” ifadesi değerlendirilir ve buradaki ifadeye göre ilgili kod bloğu çalıştırılır. Genellikle n bir değişkendir. Switch altındaki bloklar bu ifadeyle eşleşiyorsa (case) ilgili kodlar çalıştırılır. Case bloklarının sonundaki “break” ifadesi diğer bloklara geçmeyi engeller. Herhangi bir eşleşme bulunamazsa default ifadesinin altındaki kodlar çalıştırılır.

ÖRNEK

```
<?php  
$renk= "kırmızı";  
  
switch ($renk) {  
    case "kırmızı":  
        echo "Renginiz kırmızı!";  
        break;  
    case "mavi":  
        echo "Renginiz mavi!";  
        break;  
    case "yeşil":  
        echo "Renginiz yeşil!";  
        break;  
    default:  
        echo "Renginiz kırmızı, mavi ya da yeşil değil!";  
}  
?>
```

Döngüler

Koşul sağlandığı sürece aynı işi defalarca yapmak için döngü kullanılır. Döngü kurulduğu zaman döngü içinde koşulu etkileyen bir satırın bulunması gerekir. Bu olmadığı zaman döngüde kodlar sonsuza kadar çalışacaktır. Bu duruma kısır döngü kurmak denir.

PHP'deki döngüler aşağıdaki ifadelerle kurulur;

- while: Şart sağlandığı müddetçe döngü devam eder.
- do...while: Döngü içindeki kod bir defa mutlaka çalıştırılır; tekrar çalışması için şartın sağlandığı kontrol edilir.
- for : Döngü içindeki kodlar belirli bir sayıda çalıştırılır.
- Foreach : Dizideki eleman sayısı kadar döngü çalışır.

while Döngüsü

Döngü şart sağlandığı müddetçe çalışır.

SÖZ DİZİMİ

```
while (Doğrulanın Koşul) {  
    Döngü içindeki kodlar;  
}
```

Aşağıdaki örnekte değişkene (\$x) 1 sayısı atanıyor.(\$x=1) Döngü \$x değişkeni 5'den küçük ya da eşit olduğu müddetçe çalışıyor. \$x değişkeninin değerinin \$x++ parametresiyle birer artırıldığına dikkat edin.

ÖRNEK

```
<?php  
    $x = 1;  
    while ($x <= 5) {  
        echo "Say: $x <br>";  
        $x++;  
    }  
?>
```

do...while döngüsü

Bu döngü içine yazılan kodlar mutlaka bir defa çalıştırılır. Koşu kontrol edilir. Sağlanıyorsa kod çalıştırılmaya devam eder.

SÖZ DİZİMİ

```
do {  
    Döngü içindeki kodlar;  
} while (Doğrulanın Koşul);
```

Aşağıdaki örnekte \$x değişkenine 1 değeri atanıyor (\$x=1). Döngünün içine giriliyor çıktı veriliyor. \$x değişkeninin değeri bir artırılıyor (\$x++). Döngünün sonundaki parametre \$x değişkeninin değerinin 5'e eşit mi yoksa küçük mü olduğunu test ediyor. Şart sağlandığı için döngüye devam ediliyor.

ÖRNEK

```
<?php
    $x = 1;
    do {
        echo "Say: $x <br>";
        $x++;
    } while ($x <= 5);
?>
```

Bu döngünün diğerlerinden farkı en az bir defa mutlaka kodların çalıştırılıyor olmasıdır. Yani daha kodların en başında \$x değerinin değeri 10 (on) olarak atansaydı bile yine döngü içindeki kodlar bir defa çalışacaktı.

ÖRNEK

```
<?php
    $x = 10;
    do {
        echo "Say: $x <br>";
        $x++;
    } while ($x <= 5);
?>
```

for Döngüsü

İçindeki kodların kaç defa çalışacağını kesin olarak belirtildiği döngü türüdür.

SÖZ DİZİMİ

```
for (sayaç değişken; sayaç değişken koşulu; sayaç değişkenin değişimi) {
    Çalışan kodlar;
}
```

Parametreler

- Sayaç değişken: Döngünün kaç defa yapılacağını belirleyen değer başlangıcının atanması
- Sayaç değişken koşulu: Döngünün ne zaman sonlanacağını belirtildiği koşul. Koşul sağlanıyorsa döngü devam eder.
- Sayaç değişkenin değişimi: Döngü her döndüğünde sayaçta meydana gelmesi istenen değişim.

Aşağıdaki örnekte 0'dan 10'a kadar sayılar görüntüleniyor.

ÖRNEK

```
<?php
    for ($x = 0; $x <= 10; $x++) {
        echo "Say: $x <br>";
    }
?>
```

foreach Döngüsü

Bu döngü sadece dizi değişkenlerde kullanılır. Dizinin her anahtar/değeri için kullanılır.

SÖZ DİZİMİ

```
foreach ($dizi as $deger) {  
    Çalıştırılan kodlar;  
}
```

Her tekrarda dizi içindeki elemanlar \$değer ismindeki değişkene atanır. Bu işlem son eleman \$değer değişkenine atanana kadar devam eder.

Aşağıdaki örnek bir dizinin elemanlarının sırasıyla nasıl görüntülenebileceğini gösteriyor.

ÖRNEK

```
<?php  
    $renkler = array("kırmızı", "yeşil", "mavi", "sarı");  
    foreach ($renkler as $deger) {  
        echo "$deger <br>";  
    }  
?>
```

Fonksiyonlar

PHP'nin en güçlü yönü fonksiyonlardır. Sırf kendi içinde 1000'den fazla fonksiyon barındırır.

Kullanıcı Tanımlı Fonksiyonlar

Kendi içinde tanımlı fonksiyonların yanı sıra kendi tanımladığınız fonksiyonları da PHP içinde kullanabilirsiniz.

- Fonksiyon program içinde defalarca kullanabileceğiniz bir kod bloğudur.
- Fonksiyon sayfa çağrılır çağrılmaz çalıştırılmaz.
- Fonksiyonu ancak program içinde bir yerden çağırdığınızda kullanabilirsiniz.

Kullanıcı Tanımlı Fonksiyon Oluşturma

Kullanıcı tanımlı fonksiyon bildirimini "function" kelimesi kullanılarak yapılır.

SÖZ DİZİMİ

```
Söz Dizimi  
function FonksiyonAdi(){  
    Çalıştırılan kodlar;  
}
```

Fonksiyon adlarının içinde Türkçe karakter bulunmaz, mutlaka bir harfle başlamalıdır, sayı içerebilir, özel karakterler bulunmaz. Büyük/küçük harf duyarlı değildir.

Aşağıdaki örnekte "mesajYaz()" isimli bir fonksiyon oluşturuyoruz. Süslü parantezler "{", "}" fonksiyonun başladığı ve bittiği yeri gösteriyor. Fonksiyonun çıktısı "Merhaba Dünya".

ÖRNEK

```
<?php
function mesajYaz () {
    echo "Merhaba Dünya!";
}

mesajYaz (); // fonksiyon çağrılıyor.
?>
```

Fonksiyon Argümanları

Fonksiyonun için değer atamak için argümanlar kullanılır. Argüman değişken gibi düşünülmelidir. Argümanlar fonksiyon tanımlanırken kullanılan parantezler içinde belirtilir. Birden fazla argüman kullanılacaksa argümanların arasına virgül konur.

Aşağıdaki örnekte fonksiyon içinde tek bir argüman (\$sadi) kullanılmıştır. Bir isimle birlikte (Örneğin Ali) tanımlanan Soyad() fonksiyonu çağrıldığında, isimler için soyadla birlikte görüntülenecektir.

ÖRNEK

```
<?php
function Soyad($sadi) {
    echo "$sadi AK.<br>";
}

Soyad("Ali");
Soyad("Ayşe");
Soyad("Veli");
Soyad("Burak Zahit");
Soyad("Züleyha");
?>
```

Aşağıdaki örnekte iki argüman kullanılmıştır. (\$sadi ve \$yil)

ÖRNEK

```
<?php
function Soyad($sadi, $yil) {
    echo "$sadi AK. $yil yılında doğmuştur. <br>";
}

Soyad("Ali", "1975");
Soyad("Veli", "1978");
Soyad("Burak Zahit", "1983");
?>
```

Varsayılan Argüman Değeri

Bir fonksiyon şayet argümansız çağrılırsa fonksiyonun döndüreceği değer önceden belirlenebilir. Aşağıdaki örnekte bu tür kullanımı görmekteyiz.

ÖRNEK

```
<?php
function Yukseklik($enkucuk = 50){
    echo "Yükseklik: $enkucuk <br>";
}
$enkucuk(350)
$enkucuk() // fonksiyon varsayılan değer 50'yi kullanır.
$enkucuk(135)
$enkucuk(80)
?>
```

Fonksiyon İçinden Döndürülen Değerler

Fonksiyon içinden değer getirmek için "return" ifadesi kullanılır.

ÖRNEK

```
<?php
function toplama($x, $y) {
    $z = $x + $y;
    return $z;
}

echo "5 + 10 = " . toplama(5, 10) . "<br>";
echo "7 + 13 = " . toplama(7, 13) . "<br>";
echo "2 + 4 = " . toplama(2, 4);
?>
```

Diziler

Diziler çok sayıda değeri tek bir değişkende saklamak için kullanılır.

ÖRNEK

```
<?php
$arabalar = array("Volvo", "BMW", "Toyota");
echo "Araba modelleri " . $arabalar[0] . ", " . $arabalar[1] . " ve "
. $arabalar[2] . ".";
?>
```

An itibariyle birden fazla değeri tutabilen özel değişkenler dizi olarak tanımlanır. Örneğin dizi değişken kullanmadan sadece değişkenlerle araba markalarını tutmak isteseydik aşağıdaki gibi bir atama yapacaktık;

```
$araba1 = "Volvo";
```

```
$araba2="BMW";
```

```
$araba3="Toyota";
```

Yukarıdaki atama dizilere göre çok da kullanışlı değildir. Örneğin bir döngüyle özel bir araba markasını bulmak isteyebilirdik. Ya da yukarıdaki gibi üç araba markası yerine 1000 tane araba markası atamamız gerebilirdi. Bu durumda dizileri kullanmalıydık.

Bir değişken ismiyle birden fazla değeri tutmamızı sağlayan dizi içinde indeks numarasını girerek herhangi bir değere ulaşabiliriz.

Dizi Oluşturmak

PHP'de üç farklı türde dizi bulunur.

- İndeksli diziler (Indexed Arrays) – Nümerik indeksli diziler
- Çağrışımsal diziler (Associative Arrays) – Adlandırılmış anahtarlı diziler
- Çok boyutlu diziler (Multidimensional Arrays) – Birden fazla dizi içeren diziler

İndeksli Diziler

İndeksli dizi oluşturmanın iki yolu bulunur. Aşağıdaki örnekte indeks numaralarının otomatik olarak atandığı bir dizi tanımlamanın nasıl yapıldığını görüyorsunuz.

```
Şarabalar = array("Volvo", "BMW", "Toyota")
```

İndeks numaralarının belirlenerek girildiği tanımlama işlemi aşağıdaki gibi yapılır.

```
Şarabalar[0]= "Volvo";
```

```
Şarabalar[1]= "BMW";
```

```
Şarabalar[2]= "Toyota";
```

Bir dizi aşağıdaki örnekteki gibi oluşturulup ekranda çıktısı gösterilebilir.

ÖRNEK

```
<?php
    Şarabalar = array("Volvo", "BMW", "Toyota");
    echo "Araba modelleri " . Şarabalar[0] . ", " . Şarabalar[1] . " ve
    " . Şarabalar[2] . ".";
?>
```

Dizinin Uzunluğunu Almak – count() fonksiyonu

Dizilerle çalışırken count() fonksiyonu dizinin uzunluğunu (dizi elemanlarının sayısını) anlamak için kullanılır.

ÖRNEK

```
<?php
    Şarabalar = array("Volvo", "BMW", "Toyota");
    echo count(Şarabalar);
?>
```

İndeksli Dizilerde Döngü

İndeksli dizilerde saklanan değerlerle çalışırken genellikle döngüler kullanılır. Örneğin for döngüsünü bir dizinin sakladığı değerleri görmek için aşağıdaki gibi kullanabiliriz.

ÖRNEK

```
<?php
$şarabalar = array("Volvo", "BMW", "Toyota");
$uzunluk = count($şarabalar);

for($x = 0; $x < $uzunluk; $x++) {
    echo $şarabalar[$x];
    echo "<br>";
}
?>
```

Çağrışimsal Diziler – (Associative Arrays)

Çağrışimsal Diziler isimlendirilmiş anahtarlara atama yapılarak tanımlanır.

Çağrışimsal dizi oluşturmanın iki yolu bulunur. Birinci yol aşağıdaki gibidir.

```
$yas = array("Ali"=>"35", "Ahmet"=>"37", "Veli"=>"43");
```

İkinci çağrışimsal dizi oluşturma yolu aşağıdaki gibidir;

```
$yas['Ali'] = "35";
```

```
$yas['Ahmet'] = "37";
```

```
$yas['Veli'] = "43";
```

Çağrışimsal dizideki adlandırılmış adlar aşağıdaki gibi kullanılabilir.

ÖRNEK

```
<?php
$yas = array("Ali"=>"35", "Ahmet"=>"37", "Veli"=>"43");
echo "Ali " . $yas['Ali'] . " yaşındadır.";
?>
```

Çağrışimsal Dizilerde Döngü

Çağrışimsal dizilerdeki değerlerle çalışmak için foreach döngüsünü kullanabilirsiniz.

ÖRNEK

```
<?php
$yas = array("Ali"=>"35", "Ahmet"=>"37", "Veli"=>"43");

foreach($yas as $x => $deger) {
    echo "Anahtar=" . $x . ", Değer=" . $deger;
    echo "<br>";
}
?>
```

Dizilerde Sıralama

PHP’de diziler sıralanırken aşağıdaki fonksiyonlar kullanılır. Sıralama dizi içindeki elemanların yerini değiştirecektir.

- sort() – diziyi artan sıralar
- rsort() – diziyi azalan sıralar
- asort() – çağrışımsal dizilerde değerlere göre artan sıralar
- ksort() – çağrışımsal dizilerde anahtarlara göre artan sıralar
- arsort() – çağrışımsal dizilerde değerlere göre azalan sıralar
- krsort() – çağrışımsal dizilerde anahtarlara göre azalan sıralar

Dizilerde Artan Sıralama – sort()

Aşağıdaki örnekte bir dizi içindeki değerler alfabetik olarak artan sıralanıp ekranda gösterilmektedir.

ÖRNEK

```
<?php
$arabalar = array("Volvo", "BMW", "Toyota");
sort($arabalar);

$uzunluk = count($arabalar);
for($x = 0; $x < $uzunluk; $x++) {
    echo $arabalar[$x];
    echo "<br>";
}
?>
```

Aşağıdaki örnek sayılardan oluşan bir dizinin artan nasıl sıralandığını gösteriyor.

ÖRNEK

```
<?php
$sayilar= array(4, 6, 2, 22, 11);
sort($sayilar);

$diziboyutu = count($sayilar);
for($x = 0; $x < $diziboyutu; $x++) {
    echo $sayilar[$x];
    echo "<br>";
}
?>
```

Dizilerde Azalan Sıralama – rsort()

Aşağıdaki örnekte arabalar dizisi içindeki değerler alfabetik olarak azalan sıralanmıştır.

ÖRNEK

```
<?php
$arabalar = array("Volvo", "BMW", "Toyota");
rsort($arabalar);

$uzunluk = count($arabalar);
for($x = 0; $x < $uzunluk; $x++) {
    echo $arabalar[$x];
    echo "<br>";
}
?>
```

Dizi içindeki sayılar aşağıdaki örnekte azalan sıralanmıştır.

ÖRNEK

```
<?php
    $sayilar= array(4, 6, 2, 22, 11);
    rsort($sayilar);

    $diziboyutu = count($sayilar);
    for($x = 0; $x < $diziboyutu; $x++) {
        echo $sayilar[$x];
        echo "<br>";
    }
?>
```

Çağrışımsal Dizilerin Değere Göre Artan Sıralaması

Aşağıdaki örnekte çağrışımsal bir dizi atanmış değerlere göre artan sıralanmıştır.

ÖRNEK

```
<?php
    $yas = array("Veli"=>"35", "Ahmet"=>"43", "Ali"=>"37");
    asort($yas);

    foreach($yas as $x => $x_value) {
        echo "Anahtar=" . $x . ", Değer=" . $x_value;
        echo "<br>";
    }
?>
```

Çağrışımsal Dizilerin Anahtara Göre Artan Sıralaması

Aşağıdaki örnekte çağrışımsal bir dizi barındırdığı anahtarlara göre artan sıralanmıştır.

ÖRNEK

```
<?php
    $yas = array("Veli"=>"35", "Ahmet"=>"43", "Ali"=>"37");
    ksort($yas);

    foreach($yas as $x => $x_value) {
        echo "Anahtar=" . $x . ", Değer=" . $x_value;
        echo "<br>";
    }
?>
```

Çağrışımsal Dizilerin Değere Göre Azalan Sıralaması

Çağrışımsal diziler atanmış değere göre aşağıdaki gibi azalan sıralanabilir.

ÖRNEK

```
<?php
$yas = array("Veli"=>"35", "Ahmet"=>"43", "Ali"=>"37");
arsort($yas);

foreach($yas as $x => $x_value) {
    echo "Anahtar=" . $x . ", Değer=" . $x_value;
    echo "<br>";
}
?>
```

Çağrışımsal Dizilerin Anahtara Göre Azalan Sıralaması

Çağrışımsal dizilerde anahtarlara göre azalan sıralama aşağıdaki örneğe benzer şekilde yapılır.

ÖRNEK

```
<?php
$yas = array("Veli"=>"35", "Ahmet"=>"43", "Ali"=>"37");
krsort($yas);

foreach($yas as $x => $x_value) {
    echo "Anahtar=" . $x . ", Değer=" . $x_value;
    echo "<br>";
}
?>
```

Küresel (Global) Değişkenler - Süper Küreseller (Superglobals)

Süper küresel değişkenler, betiğin içindeki herhangi bir kısımda, fonksiyon içinde ya da sınıfta herhangi özel bir şey yapmaksızın erişilebilen ön tanımlı değişkenlerdir.

PHP içindeki süper küresel değişkenler şunlardır;

- \$GLOBALS
- \$_SERVER
- \$_REQUEST
- \$_POST
- \$_GET
- \$_FILES
- \$_ENV
- \$_COOKIE
- \$_SESSION

\$GLOBALS

PHP betiği içinde herhangi bir yerden ulaşılabilen değişkenlerdir. (Fonksiyonlar ve metodların içinden dahi ulaşılabılır.) Bütün değişkenler ismi \$GLOBALS[indeks] olan bir dizi içinde saklanır. İndeks ifadesi değişkenin ismidir.

Aşağıdaki örnek \$GLOBALS anahtarının nasıl kullanılabileceğini gösteriyor.

ÖRNEK

```
<?php
    $x = 19;
    $y = 10;

    function ekle () {
        $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
    }
    ekle ();
    echo $z;
?>
```

\$x, \$y değişkenlerine normal şartlarda fonksiyon içinden ulaşamazken artık ulaşılabilir olduğuna, \$z değişkeninin değerine de fonksiyon dışından ulaşılabilmesine dikkat edin.

\$_SERVER

Başlık, yol ve betiğin bulunduğu yer gibi bilgilerin tutulması için \$_SERVER küresel değişkeni kullanılır.

Aşağıda \$_SERVER küresel değişkeninin nasıl kullanılabileceğini görüyorsunuz.

ÖRNEK

```
<?php
    echo $_SERVER['PHP_SELF'];
    echo "<br>";
    echo $_SERVER['SERVER_NAME'];
    echo "<br>";
    echo $_SERVER['HTTP_HOST'];
    echo "<br>";
    echo $_SERVER['HTTP_REFERER'];
    echo "<br>";
    echo $_SERVER['HTTP_USER_AGENT'];
    echo "<br>";
    echo $_SERVER['SCRIPT_NAME'];
?>
```

Aşağıdaki tablo \$_SERVER küresel değişkeni ile ulaşılacak bilgilerin genel bir listesini sunmaktadır.

Kod	Açıklama
\$_SERVER['PHP_SELF']	O an çalıştırılan betiğin dosya adını döndürür.
\$_SERVER['GATEWAY_INTERFACE']	Serverın kullandığı CGI işleyicinin versiyon numarasını döndürür.
\$_SERVER['SERVER_ADDR']	Sunucunun IP adresini döndürür.
\$_SERVER['SERVER_NAME']	Sunucu bilgisayarın adını döndürür.
\$_SERVER['SERVER_SOFTWARE']	Web Sunucu yazılımının tanımlama metnini döndürür. (Apache, IIS vs.)
\$_SERVER['SERVER_PROTOCOL']	http revizyon numarasını döndürür.
\$_SERVER['REQUEST_METHOD']	Formdan gelen verinin istek yöntemini döndürür (POST, GET)
\$_SERVER['REQUEST_TIME']	İsteğin başlangıç zaman damgasını döndürür.
\$_SERVER['QUERY_STRING']	Sayfaya query string ile ulaşılmışsa query string satırını döndürür.
\$_SERVER['HTTP_ACCEPT']	Mevcut isteğin Kabul Başlığını (Accept header) döndürür.

\$_SERVER['HTTP_ACCEPT_CHARSET']	Mevcut isteğin dil kodlamasını döndürür. (utf-8, ISO-8859-1)
\$_SERVER['HTTP_HOST']	Mevcut isteğin sunucu başlık bilgisini döndürür.
\$_SERVER['HTTP_REFERER']	O anki sayfanın tam URL'sini döndürür.
\$_SERVER['HTTPS']	Betiğin güvenli http protokolü üzerinden çalışıp çalışmadığını döndürür.
\$_SERVER['REMOTE_ADDR']	Tarayıcının çalıştığı istemcinin IP adresini döndürür.
\$_SERVER['REMOTE_HOST']	Tarayıcının çalıştığı istemci bilgisayarın adını döndürür.
\$_SERVER['REMOTE_PORT']	İstemci bilgisayarın sunucusuyla bağlı olduğu IP portunu döndürür.
\$_SERVER['SCRIPT_FILENAME']	Çalıştırılan betiğin bulunduğu yol bilgisini döndürür.
\$_SERVER['SERVER_ADMIN']	Web sunucusu konfigürasyon dosyasına yazılan SERVER_ADMIN direktif bilgilerini döndürür.
\$_SERVER['SERVER_PORT']	Web sunucusunun iletişim için kullandığı portu döndürür.
\$_SERVER['SERVER_SIGNATURE']	Sunucu bilgisayar tarafından üretilen sunucu versiyonu ve sanal sunucu ismini döndürür.
\$_SERVER['PATH_TRANSLATED']	O an çalışan betiğin sistem tabanlı dosya yolunu döndürür.
\$_SERVER['SCRIPT_NAME']	O an çalışan betiğin çalıştığı yolu döndürür.
\$_SERVER['SCRIPT_URI']	Betiğin çalıştırıldığı sayfanın URI bilgisini döndürür.

\$_REQUEST

HTML formu tarafından gönderilen bilginin toplanması için kullanılır.

Aşağıdaki örnekte <form> etiketleri içinde isim bilgisini aldığımız bir alan ve bunu yollamak için kullandığımız bir düğme html kodlarıyla oluşturulmuş bulunuyor. Örneğimiz kendi topladığı veriyi kullanabilen bir dosyadan oluşuyor. Kullanıcı Gönder düğmesine bastığında şayet post metodu ile veri gelmişse php kodları gelen datayı \$_REQUEST ile topluyor. Koşullu ifade gelen verinin durumuna göre "İsim alanı boş" ya da girilen ifadeyi yazıyor.

ÖRNEK

```
<html>
<body>

<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
İsim: <input type="text" name="isim">
<input type="submit">
</form>

<?php
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        // Input alanının değeri alınıyor.
        $ad= $_REQUEST['isim'];
        if (empty($ad)) {
            echo "İsim alanı boş";
        } else {
            echo $ad;
        }
    }
?>

</body>
</html>
```

\$_POST

Html formundan post yöntemiyle gönderilen veriler toplanırken en yaygın kullanım \$_POST yöntemidir.

Aşağıdaki örnekte <form> etiketleri içinde isim bilgisini aldığımız bir alan ve bunu yollamak için kullandığımız bir düğme html kodlarıyla oluşturulmuş bulunuyor. Örneğimiz kendi topladığı veriyi kullanabilen bir dosyadan oluşuyor. Kullanıcı Gönder düğmesine bastığında şayet post metodu ile veri gelmişse php kodları gelen datayı \$_POST ile topluyor. Koşullu ifade gelen verinin durumuna göre “İsim alanı boş” ya da girilen ifadeyi yazıyor.

ÖRNEK

```
<html>
<body>

<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
İsim: <input type="text" name="isim">
<input type="submit">
</form>

<?php
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        // Input alanının değeri alınıyor.
        $ad= $_POSTD['isim'];
        if (empty($ad)) {
            echo "İsim alanı boş";
        } else {
            echo $ad;
        }
    }
?>

</body>
</html>
```

\$_GET

Html formundan “GET” yöntemiyle gönderilen veriler \$_GET ifadesiyle toplanır. \$_GET aynı zamanda adres çubuğuyla (ya da URL) gönderilen verileri toplamak için kullanılır.

Herhangi bir html dosyasına aşağıdaki kodu girin;

ÖRNEK

```
<html>
<body>
<a href="denemeget.php?konu=PHP&okul=IIBF">Deneme</a>
</body>
</html>
```

Kullanıcı linki tıkladığında (Deneme) “konu” ve “okul” parametreleri aşağıda kodları bulunan “denemeget.php” dosyasına gönderilecektir. Gelen URL kodu \$_GET ile işlenecektir.

ÖRNEK

```
<html>
<body>

<?php
    echo "Ders " . $_GET['konu'] . " ve burası " . $_GET['okul'];
?>

</body>
</html>
```