PHP PANDAS

HERKES İÇİN PHP PROGRAMLAMA DİLİ

DAYLE REES & HASAN DEĞİŞMEZ

PHP Pandas (TR)

Dayle Rees ve Hasan Degismez

Bu kitap http://leanpub.com/php-pandas-tr adresinde satıştadır.

Bu versiyon, 2019-06-29 tarihinde yayınlanmıştır

İçindekiler

Teşekkürler . . . . . **i**

Hatalar . . . . . **ii**

Geri Bildirim . . . . . **iii**

Çeviriler . . . . . **iv**

Giriş **1**

Yükleme **3**
Linux 3
Mac OSX 4
Windows 5

Cevapları Bulmak **7**
Geliştiriciler robottur. 7
Google kullanma sanatı 8

Dosyalar **10**

Temel Aritmetik **12**
İfadeler 12
Aritmetik Operatörler 14
Prosedür 15

Değişkenler & Atama **19**
Minik Kutucuklar 19
Tam benim tipim 21
Gelişmiş Atama 23

Teşekkürler

Herşeyden önce kız arkadaşım Emma'ya teşekkür ediyorum, sadece benim inek şakalarıma katlandığı için değil, ayrıca iki kitabım için inanılmaz kırmızı panda fotoğrafları çektiği için! Seni seviyorum Emma!

Benim matematik kutularına ilgimi 30 yıl boyunca destekleyen aileme teşekkür ediyorum! Ayrıca kitabın milyarlarca kopyasını ilk olarak aile bireylerine aldıkları için!

JustPark'da benden desteğini esirgemeyen tüm harika iş arkadaşlarıma teşekkür ediyorum! Siz inanılmazsınız!

Daha önce yazdığım Code Happy ve Code Bright kitaplarını alan herkese ve tüm Laravel topluluğuna teşekkürler. Sizin desteğiniz olmasaydı asla yazmaya devam edecek cesareti kendimde bulamazdım.

Çevirenin Notu

Dayle Rees gerçekten bazen sıkıcı olan konuları eğlenceli bir şekilde yazmayı iyi beceriyor. 10 senenin üzerinde PHP kodu geliştirme deneyimine sahip geliştiricilerin size bu basitlikte bir kitabı sunması gerçekten sabır ve kararlılık gerektiriyor.

Her zaman yanımda olan sevgili eşim Hatice'ye ve minik pıtırcığım Nil'e teşekkür ediyorum.

Bu kitabın çevirisinde ve hata düzeltmelerinde yardımcı olan Alper Kaya'ya teşekkür ediyorum.

Çevirilerde Türkçe'ye olabildiğince uyarlamaya çalıştım, görebildiğiniz bir hata veya hoşunuza gitmeyen bir yer olursa lütfen güncellenmesi için bana iletirseniz çok sevinirim.

Twitter: http://twitter.com/hasandz[1]

İnternet: http://hasan.degismez.com[2]

[1] http://twitter.com/hasandz
[2] http://hasan.degismez.com

Hatalar

Bu belki benim üçüncü kitabım ve yazma becerim bir öncekine göre biraz daha gelişmiş olabilir, ama sizi temin ederim kitapta çok fazla hata bulunacaktır.

Kitabı desteklemek için lütfen bulduğunuz hataları konu başlığı ile birlikte bana e-posta olarak gönderiniz: me@daylerees.com[3].

Hatalar bulunduğunda düzeltilecektir. Konuların düzenlenmiş hali bu kitabın gelecek güncellemelerinde yayınlanacaktır.

[3] mailto:me@daylerees.com

Geri Bildirim

Aynı şekilde, eğer kitap veya içerikle ilgili geri bildirim yapmak isterseniz lütfen bana ulaşın. Email adresime (me@daylerees.com[4]) gönderebilir veya twitter üzerinden @daylerees'e tweet atabilirsiniz.

Gelen tüm mesajları cevaplamak için tüm gayretimi göstereceğim.

[4] mailto:me@daylerees.com

Çeviriler

Eğer PHP Pandas kitabını kendi dilinize çevirmek isterseniz lütfen me@daylerees.com[5] adresine düşüncelerinizi mail atın.

Lütfen bu kitabın markdown formatında yazıldığını unutmayınız.

[5] mailto:me@daylerees.com

Giriş

Merhabalar! Sen varya sen, dünya üzerindeki en yakışıklı/güzel okuyucusun! PHP Pandas kitabını satın aldığın için ve meşhur bir internet geliştiricisi olma yolunda ilk adımı attığın için tebrikler.

Ben kim miyim? Bu oldukça basit bir soru! Benim adım Dayle ve bu macerada senin yazarın olacağım. Bir kaç yıldır yeni başlayanlar için kitaplar yazıyorum ve senin gibi çekici bir çok okuyucu ile yeni beceriler elde ettikleri maceralara katıldım. Birlikte yeni keşifler yapacağız ve şundan emin olabilirsiniz ki yolculuk boyunca senin yanında olacağım.

> Neden bir deli gibi yazıyorsun?

Pardon? He şu.. Görebileceğiniz üzere, bu bildiğim tek yazım şekli. Eğer tamamen teknik bir kitap ve katı bir fen bilgisi hocası tarzında bir yazım bekliyorsanız, o zaman üzgünüm yanlış yere geldiniz. Ben kitaplarımı insanlar için yazıyorum. Ben arkadaş olduğumuzu düşünüyorum, bir barda oturmuş büyük bira bardaklarımızın üzerinden bakarken, PHP hakkında konuşuyormuş gibi hayal ediyorum.

Gerçek şu ki, benim hedef aldığım yeni başlayanlar bu yazım tarzını seviyor gibi. Bu kitabı okuyup üstün bir matematik derecesi elde etmeyi beklemiyorlar, bunun yerine PHP hakkında bir kaç şey öğrenmeyi amaçlıyorlar ve işte ben bunun için size söz verebilirim.

Aha, sen nasıl konuştuğumuzu fark etmiş olmalısın. Diğer yazarlarda bunu yapamıyorsun öyle değil mi? Gördün mü, özel sihirli gücüm sayesinde seni konuşturup bana soru sormanı sağlayabiliyorum.

> Bi dakika, bunu nasıl yapıyors…

Bu meslek sırrımı söylemek olur. Üzgünüm, henüz bunu paylaşamayız, ama bu maceranın bir parçası olduğun için hoşnut değil misin, sadece bir gözlemci olmadığın için yani?

> Yani sanırım… En azından bir deyeceğime emin olabilirsin.

Mükemmel.

İşte şimdi diğer tüm kitapların PHP hakkında bir şeyler anlattıkları kısma geldik, tüm geçmişi, uygulama alanları, yazarı ve milyonlarca diğer şey. Sanırım şimdiye kadar geleneksel bir yazar olmadığımı anlamışsındır ve ben bu tarz bölümlerin meraklısı değilimdir. Bu kitabı PHP öğrenmek için satın aldın, demek oluyor ki bu dil hakkında bir merakın oluşmuş. Bence bu senin için yeterli olacaktır.

PHP internet üzerindeki çoğu büyük, geniş sitenin yazıldığı bir programlama dilidir. Orjinal hali Rasmus Lerdorf adında biri tarafından yazılmış, Google'da bulduğunuz hemen hemen tüm

resimlerinde yüzünde bir gülücük olduğunu görebilirisiniz. Şimdi, Rasmus gerçekten harika bir adam ve her gün bana bu zanaatı kazandırdığı için kendimce ona teşekkür ediyorum, ama sanırım sizin onun hakkında bilmeniz gerekenler bu kadar. Diğer PHP kitapları size onun en sevdiği mısır gevreğini bile söyler muhtemelen, bunun yerine direk olarak öğrenmeye başlasak nasıl olur?

Bu kitap ***tamamen sıfırdan*** başlayanlar için. Yani daha önce hayatında hiç programlama ile haşır neşir olmadıysan, o zaman şanslısın demektir! Eğer daha önce denediysen de, sorun olmaz. Eğer bir PHP uzmanıysan, o zaman belki bildiklerini tekrar gözden geçirebilirsin ve belki de bunu yaparken bir kaç yeni numara öğrenirsin.

Kız arkadaşımı (bir programlama deneyimi yok), teknik olmayan iş arkadaşlarımı ve sokaktaki rastgele insanları kobay faresi olarak kullanıp, zorla kitabımı okutuyorum, böylece PHP hakkında hiç bilgisi olmayan insanların nasıl ilerlediğini inceliyorum. Kobay farelerim oldukça iyi işler çıkardı, şimdi sıra sende!

Bu kitabı yazarken hedefim, piyasadaki en eğlenceli, eksiksiz ve harika PHP kitabı olması. Birisi PHP öğrenmek istediğinde önerilen **o kitap** olmasını istiyorum. Herkese yönelik olabilmesi için çok çalıştım, bu yüzden sende bu maceradan keyif alırsan lütfen tweet'le, bloğuna yaz, arkadaşların ve ailen için bir kaç kopyasını satın al, yada sadece yazıcıdan çıkartıp sokaktan geçen insanların suratına yapıştır.

Bu kitap PHP'nin nasıl yazıldığını anlatan bir kitaptır. Size nasıl internet siteleri yapacağınızı öğretmez (Bu konuyla ilgili olarak seriye yeni bir kitap eklemek için çalışıyorum). Bunun yerine, ilk internet sitenizi yaparken kullanacağız dil ile ilgili bilgi dağarcığı oluşturacak ilk adım olarak düşünebilirsiniz, çok s@%£^ olacaksın bebeğim!

Eğer bu kitabı okurken bir şeyin eksik olduğunu hissedersen, bir bölüm karmaşık geldiyse veya canını sıkan herhangi bir şey varsa, lütfen beni `me@daylerees.com` adresine eposta yollayarak haberdar et. İnanılmaz biçimde geri dönüş yaparım (sağolsun tüm medya sorgularım… haha… programcı şakası) ve bu kitabın herkes için mükemmel olmasını istiyorum.

Eğer bu kitabı okudun ve hiçbir sorun bulamadıysan, iyi işte… bana bir eposta gönder ve beğendiğini söyle! Bunu senden duymayı çok isterim.

Tamam öyleyse, daha fazla vakit kaybetmeyelim. Yeni beceriler öğrenmek üzeresin! Sayfayı çevir, Jurassic Park'ta kapıların açıldığı anı hayal et ve programcılık dünyasına girmeye hazır ol!

Yükleme

PHP ile çalışmaya başlamadan önce, onu bilgisayarımıza yüklemeliyiz. Göreceğin üzere PHP diğer normal bir uygulama gibidir. PHP kodlarını çalıştırabilmesi için bilgisayarınıza yüklenmesi gerekir.

Yükleme metodları işletim sistemine göre büyük farklılıklar gösterir. Bu yüzden PHP yükleme için sana üç farklı yol haritası sundum. İlk bölüm sana Linux üzerinde PHP yüklemeyi açıklıyor, dağıtım popülerliğinden dolayı Ubuntu seçildi. İkinci bölüm Apple Mac OSX sistemi üzerinde PHP kurulumunu açıklıyor. Son olarak, üçüncü bölüm Windows işletim sistemi üzerinde PHP kurulumunu açıklıyor.

PHP'nin sadece konsol versiyonunu yükleyeceğiz. Henüz bir ağ sunucusu kurulumu yapmayacağız. Bu konuya ilerleyen bölümlerde değineceğiz. Konsol versiyonu PHP öğrenmeye başlamak için ihtiyacımız olan herşeye sahip.

Unutma, sadece kendi bilgisayarınla ilgili bölümü okuman yeterli. Bir kere PHP yüklendikten sonra, bu bölümü atlayıp kitabın diğer bölümüne geçebilirsin.

Linux

Unix tabanlı bir Linux dağıtımında PHP kurmanın en iyi yolu paket yönetim sistemini kullanmaktır. Paket yönetim sistemi büyük oranda kullandığın dağıma bağlı olarak değişir. Ben sana en popüler dağıtımlardan biri olan Ubuntu üzerinde adımları anlatmaya karar verdim.

Ubuntu paketlerini kurmak için `apt` paket yönetim sistemini kullanmakta. PHP'nin konsol versiyonunu kurmak için `php5-cli` paketini kurmamız gerekiyor. Hadi hemen bunu yapalım. İlk önce yeni bir konsol açın. Sonra aşağıdaki satırı buraya yaz.

```
1  $ sudo apt-get install php5-cli
```

Buradaki dolar işaretini yazman gerekmiyor, bu sadece bizim komutu konsola yazdığımızı belirten bir uç birim işareti. Giriş(Enter) tuşuna bastığında, apt senin için PHP uygulama paketlerini indirecek ve kuracaktır.

İşte bu kadar! İşin bitti, yani öyle olması lazım. Haydi kontol edelim, ne dersiniz? Basitçe şunu yazabilirsin..

```
$ php -v
```

Bu komut sisteminde kurulu olan PHP versiyonunu göstermek için kullanılıyor. Aşağıdakine benzer bir çıktı görmen lazım.

```
PHP 5.5.13 (cli) (built: Jun  5 2014 19:13:23)
Copyright (c) 1997-2014 The PHP Group
Zend Engine v2.5.0, Copyright (c) 1998-2014 Zend Technologies
```

Seninki tamamen aynısı olmayacak, sonuçta hepimiz farklıyız, değil mi? Yukarıdaki örnekte PHP versiyonu `5.5.13`. Umalım ki senin PHP versiyonunun `5.4.0` veya yukarısı olsun.

Eğer senin versiyonunun doğru değilse, o zaman kullandığın Linux dağıtımının dökümanlarına başvurarak, nasıl uygun bir versiyon yüklemen gerektiğini bulman gerekiyor.

Hadi şimdi bir sonraki bölüme atla, burada işin bitti!

Mac OSX

Macintosh işletim sisteminde PHP kurulu olarak gelmektedir. Hemen Terminal uygulamasını açın ve PHP versiyonunu öğrenmek için aşağıdaki komutu yaz.

```
$ php -v
```

Dolar işaretini yazma, bu sadece bizim komutu konsola yazdığımızı belirten bir uç birim işareti. Aşağıdakine benzer bir çıktı görmen gerekiyor, fakat tamamen aynısı değil.

```
PHP 5.4.24 (cli) (built: Jan 19 2014 21:32:15)
Copyright (c) 1997-2013 The PHP Group
Zend Engine v2.4.0, Copyright (c) 1998-2013 Zend Technologies
```

Bu örnekteki PHP versiyonu `5.4.24`. PHP `5.4` versiyonundan yüksek olduğu sürece bir sorun yok, kitabın bu bölümünü atlayıp bir sonraki konuya başlayabilirsiniz.

Eğer seninki düşük bir versiyon ise, sorun değil. Biz üçüncü parti bir paket yönetim sistemini kullanarak OSX üzerine PHP'nin yeni versiyonunu kurabiliriz.

'Homebrew' veya kısaca 'Brew' olarak adlandırılan paket yönetim sistemini kullanacağız. Homebrew kurmak için aşağıdaki sitedeki adımları takip etmen gerekiyor:

brew.sh[6]

Gereki komutları ve adımları buraya kopyalamak istemiyorum, çünkü her sürüm ile birlikte bunlar değişebiliyor. Bir kere Homebrew kurulduktan sonra, artık daha yeni bir PHP versiyonu kurmanın vakti geldi demektir. Benim önerim `5.5` versiyonunu kurmanız. Bunu aşağıdaki komutu kullanarak yapabilirsiniz.

[6] http://brew.sh/

```
$ brew install php55
```

Sonrasında bu PHP versiyonunun bulunduğu yeri sisteminin PATH değişkenine tanımlaman gerekiyor. Endişelenme, sadece aşağıdaki komutu yaz.

```
$ PATH=~/usr/local/Cellar/php55/5.5.13/bin:$PATH
```

Buradaki versiyon numarasını Homebrew'in yüklediği versiyon olarak değiştirmeniz gerekiyor. Şimdi tekrar bakıp PHP versiyonunu kontrol edelim.

```
$ php -v
```

Umuyorum ki, bu sefer PHP 5.4 üzerinde bir versiyonuna sahipsiniz. Hadi şimdi kitabın bir sonraki bölümüne geçin.

Windows

Windows üzerinde PHP kurmak biraz daha zor, en azından benim için öyle. Aşağıdaki adımları kendi Windows 10 makinemde test ettim, fakat adımları uygulamakta bir sorun yaşarsan, bunu bana bildir ve bende senin için daha iyi Windows bilen birini bulup bu bölümü baştan yazdırayım.

İlk önce, aşağıdaki adrese gidin:

http://windows.php.net/download

Burada PHP 5.4 üzerinde bir versiyonu zip dosyası olarak indirmelisin. Arşiv dosyası indikten sonra, bunu mantıklı bir klasöre açmalısın. Ben kendi klasörümü şu şekilde seçtim:

```
C:\Users\Dayle\PHP
```

Bu kitaptaki komutları çalıştırmak için bir konsola ihtiyacımız var. Windows üzerinde konsol çalıştırmanın çok iyi bir yolu var.

Masa üstünde veya herhangi bir klasör içinde sağ tıklayın, 'Kısayol Oluştur' seçeneğini seçin, açılacak kutuya şunu yazın:

```
cmd.exe
```

Sonraki tuşuna bas ve kısayoulun adını "PHP" olarak adlandır.

Son olarak, oluşturduğun kısayola sağ tıkla ve özelliklere bas. Yukarıdaki 'Kısayol' sekmesini aç, 'Başlama yeri' ile belirtilen alana PHP arşivini çıkardığın klasörün yolunu yaz. Bitirdiğinde 'Tamam' tuşuna bas.

Oluşturduğun kısa yola çift tıkla, karşına komut satırı çıkacak. Şunu yaz...

```
1 php -v
```

..ve PHP versiyon bilgisi karşına çıkıyor olmalı. Bu versiyonun PHP 5.4 üzerinde olduğunu onayladıktan sonra, kitabın bir sonraki bölümüne geçebilirsin.

Bir kez daha, bu bölümün gelişi güzelliği için özür diliyorum. Yazılım geliştirmek için Windows bir makineyi kullanmayalı yıllar oluyor. Eğer içinizden biri Windows üzerinde PHP'nin daha iyi çalıştırılabileceğini biliyorsa, lütfen talimatları e-posta olarak yollasın ve 5 dakikalık şöhrete bu konu başlığı altında sahip olsun!

Cevapları Bulmak

Biliyorum. Bu biraz havada kalıyor öyle değil mi? Ben sana bunun önemli bir konu olduğunu söylediğimde bana güvenmelisin. Bu bölüm senin gibi yeni bir geliştiricinin kendine güvenmesini amaçlıyor. Öğrenmek zor, fakat endişe etme; Bu süreçte sana yardımcı olacağım.

Geliştiriciler robottur.

Neden bir geliştirici olmaya karar verdin? Tamam, durun! Tahmin edeyim. New York'un en gözde gece mekanlarından birinde limuzininden kasıla kasıla inen bir PHP geliştirici gördün, 5 şişe Cristal sipariş etti ve gecenin kalanını Jay-Z ve Tupac'ın hayaletiyle takılarak geçirdi.

Evet doğru, geliştiricilerin hayatı oldukça şatafatlı. Bu sayfaları gün içindeki 5 saat ayık kaldığım zaman diliminde yazmak zorundayım. Muhtemelen bir geliştiricinin kod yazdığını gördün ve şöyle düşündün…

> Hadi canım, bu geliştirici bir robot olmalı. Tüm bu kodların içindeki kelimeleri, fonksiyonları ve hepsinin nasıl çalıştığını biliyor.

Geliştiricilik tecrübesine sahip olmayanlar, geliştiricilerin birer dahi olduklarını ve matematik alanında üstün başarılı olduklarını düşünebilir. Belki bazı geliştiriciler için bu doğru olabilir, ama kesinlikle benim için doğru değil. Sanırım bir çok geliştirici de burada bana katılacaktır.

Gerçek şu ki, biz mükemmel değiliz. Biz mükemmele yakın bile değiliz. Eğer siz geliştiricilerin tüm bu PHP fonksiyonlarını ve kod bloklarını hafızalarında tuttuklarını düşünüyorsanız, o zaman sen hiçbir zaman bu kapasiteye ulaşamayacağınızı düşünüp kendini kandırıyorsun demektir.

Bu kesinlikle doğru değil. Biz herşeyi oturup ezberlemiyoruz. Aslında günlük olarak yazdığımız kodların çoğu başka bir yerden alınma. Biz google savaşçılarıyız. Metinler üzerinde çok basit işlemler yapan PHP fonksiyonları vardır, bunlara yazmam gereken parametrelerin sırasını bile nerdeyse her hafta PHP dökümanlarından bakıp tekrar hatırlarım.

Eğer tam anlamıyla takıldıysam, Google kullanarak başka bir geliştiricininde aynı sorunlarla karşılaşıp karşılaşmadığına bakarım. Çoğunlukla başka bir geliştiricinin uyguladığı bir çözümü bulurum veya beni çözüme götürecek kadar gerekli bilgiyi bulurum. Tabiki bu süreç çift taraflı olarak işler, ben de kendi bulduğum çözümleri tekrar topluluğa bildirmeye çalışırım. Ben her zaman Stack Overflow üzerinde cevaplar yazacağım ve tartışma kanallarına katkıda bulanacağım. Bilgiyi topluluğa geri kazandırmak önemlidir.

Gördüğünüz üzere, biz robot değiliz. Biz programlama dili ile ilgili herşeyi bilmiyoruz, tüm sorunlara uygun bir çömümüz yok. Fakat biz şahane birer araştırmacıyız. Biz fırsatçılarız. Biz becerikli problem çözücüleriz. Biz geliştiricileriz.

Google kullanma sanatı

İnsanlar size bir şeyi Google'da aramanızı söylediklerinde, bunun bir hakaret olarak algılanması kolaydır. Anasayfamızın Google olmasının bir sebebi var. Hadi gelin birlikte sıklıkla karşılaşılan programlama sorunlarını nasıl çözeceğimizi öğrenelim.

Biz bir program yazıyoruz ve bir yerinde bir cümleyi ters çevirmemiz gerekiyor, böylece 'Pandalar harika!' işlemden sonra '!akirah raladnaP' olacak. Böyle bir işi nasıl yapacağımız hakkında hiçbir fikrimiz yok. PHP'ye henüz yeni başladık.

PHP'de bir dizi karaktere 'dizge' (string) denildiğini biliyoruz. Biliyoruz çünkü bu Panda örneklerinin olduğu çılgın kitaba umut bağlamadık ve bir sonraki bölümden bunu öğrendik. Öyle değil mi?

Öyleyse ne yapmak istediğimizi biliyoruz. Bir dizgeyi ters çevirmek istiyoruz. Gelin şimdi bunun için bir arama yapalım.

```
1 dizge ters çevirme
```

Dur bekle! Problem şu ki, binlerce farklı programlama dili mevcut. Cidden, bilgisayarlar uzun bir süredir aramızda.

Eğer 'dizge ters çevirme' olarak aratırsak, C++, ASP.NET, Erlang gibi daha bir çok dil için sonuçlar çıkar. Bizim odak noktamız PHP. Diğer diller umrumuzda değil. PHP'de uzman olduktan sonra diğerleriyle oynayabiliriz. Şimdi bu problemi programlama dilini de aramaya ekleyerek çözelim.

```
1 php dizge ters çevirme
```

Harika. Şimdi Google'dan dönen sonuçlara bir göz atalım. Bu arada Google için çalışmadığımı ve herhangi bir komisyon almadığımı belirmemin vakti geldi sanırım. İstiyorsanız gidip Bing kullanabiliriz, tabi sonunda dizge ters çevirme fonksiyonu yerine kendinizi kullanılmış at romörku satın alırken bulabilirsiniz. Öyleyse nerde bu sonuçlar?

Bir dizgeyi ters çevirir - PHP

http://php.net/manual/tr/function.strrev.php[7]

Reverse a string with php - Stack Overflow

http://stackoverflow.com/questions/11100634/reverse-a-string-with-php[8]

[7] http://php.net/manual/tr/function.strrev.php
[8] http://stackoverflow.com/questions/11100634/reverse-a-string-with-php

---

Doğru soruyu sorarak, sonuçta kullanışlı kaynaklar elde ettik. PHP Kılavuzu ve Stack Overflow internet üzerindeki en iyi PHP problem çözüm merkezleridir. Her zaman doğru cevaplar burada olacak demiyorum. Başka çok güzel sitelerde var, ama bir süre sonra arama sonuçlarının ne kadar çok bu iki site üzerinde yoğunlaştığını fark edeceksiniz.

Şimdi bir dizgeyi ters çevirecek bir araç arıyoruz. Burada soyut bir problemi çözmeye çalışmıyoruz, tam olarak istediğimiz şeyi biliyoruz.

Devam et ve ilk bağlantıya tıkla, `strrev()` adındaki fonksiyonun sevgili PHP kılavuz sayfası seni karşılayacak. Bir fonksiyonun nasıl kullanıldığını bilmek zorunda değilsin. Burası şimdilik kafana takılmasın.

Fonksiyonların nasıl kullanılacağını kavradığın zaman, PHP klavuz sayfasının `strrev()` fonksiyonunun kullanımıyla ilgili tüm bilgileri ve nasıl kullanılacağı ile ilgili örnekleri içerdiğini göreceksin.

Gördüğün üzere doğru soruları sorarak, işimize devam etmemiz için gerekli tüm yardımı aldık. Daha önce `strrev()` fonksiyonuyla ilgili hiçbir fikrimiz yoktu, bunun yerine çözmemiz gereken problemi biliyorduk. Bu bizi çözüme götürmek için yeterli. Bu sayfayı daha sonra tekrar ziyaret etme ihtiyacı duymamızın bir sakıncası yok.

Belkide bu fonksiyonu yeterince sık kullanmadığımız için kullanım özelliklerini hatırlamamız gerekecek. Bununla birlikte bu fonksiyonu daha çok kullanmaya başladığında ve bu kılavuz sayfasını bir çok kez ziyaret ettiğinde, bunu kolay kolay unutmazsın. Aynı problemle karşılaştığınızda hemen şu şekilde düşüneceksiniz 'Burada her zaman kullandığım `strrev()` fonksiyonunu kullanabilirim, nasıl kullanıldığını tam olarak biliyorum'. Bu bir kas hafızasına dönüşecektir ve alet çantanın bir parçası olacaktır.

Yani kitabın bu bölümünde sana öğretmeye çalıştığım ders şudur, panik yapma. Herşeyi hatırlamak zorunda değilsin ve soru sorup yardım istemek tamamen doğal bir şey. Aslında yardım istemek ve deneyimlerinden yeni şeyler öğrenmek insanca bir davranış.

Tebrikler! Sende bir insansın, bir robot değilsin.

Dosyalar

İşte seni şok edecek bir haber. PHP kodları dosyalarda tutuluyor. Özür dilerim ama bu doğru! Çok ama çok fazla dosyayla çalışacaksın. Aslında, bazen tek bir dosya, ama sonrasında çok daha fazlası olacak.

Şimdi bu şok edici gerçek yolumuzdan çekildiğine göre, bir PHP dosyası nasıl oluşturulur öğrenmenin vakti geldi.

> Dayle, ben zaten bilgisayardaki dosya sisteminin nasıl çalıştığını biliyorum.

Tebrikler dostum! Senin için iyi bir şey, ama bizim varmak istediğimiz nokta bu değil. Göreceğin üzere hemen hemen tüm PHP dosyaları ortak bir şeye sahip. PHP açılış etiketinden bahsediyorum.

Bu ufaklığa yakından bak.

Örnek 01: PHP etiketi.

```
1 <?php
```

Çok güzel değil mi? Pırıl pırıl parlıyor. Gerçekten olağanüstü bir simge.

> Ee.. Ben..

Ne? Onun hakkında benimle aynı şeyleri hissetmiyor musun? Bana güven, yıllarca PHP ile yazılım geliştirdikten sonra sende onu oldukça güzel bulacaksın. Gece gözlerini kapatıp uykuya dalarken onu göreceksin. O, senin en iyi arkadaşın olacak. O, senin PHP kullanmanı sağlayacak.

Ben her zaman pratik örnekler üzerinde ilerlemeyi tercih ederim, öyleyse hadi birlikte bir şey deneyelim. Yeni bir dosya oluştur, adı `test.php` olsun. PHP dosyaları genellikle `.php` uzantısını kullanır. Aslında bu uzantı olmadan da çalıştırabiliriz, ama buna uymak senin yararına, eğer uymazsan daha büyük geliştiriciler sana güler, öğle yemeğini çalar ve seni ağlatırlar. Şakaydı.. geliştiriciler arkadaş canlısı bir topluluk, ama yinede `.php` uzantısını kullanmalısın.

Her şeyden önce, hadi dosyanın içine…

Örnek 02: Biraz yazı.

```
1 Pandalar harika!
```

…kelimelerini yazalım, ve kaydedelim.

Harika, şimdi dosyayı çalıştıralım. Bunu kullanmak için komut satırında veya unix terminalinde `php` uygulamasını çağırmak gerekiyor, sonra buna dosya adını parametre olarak vereceğiz. Örneğin, benim Mac'imde şöyle bir şey yazıyorum.

Örnek 03: Bir PHP dosyasını çalıştırma.

```
php test.php
```

Ekrana `Pandalar harika!` yazdığını göreceksin. Bunun sebebi PHP etiketleri dışındaki her şey uygulama çalıştırıldığında ekrana basılır. Şimdi başka bir şey deneyelim. İlk PHP etiketimizi kullanacağız.

Dosyayı düzenle ve aşağıdaki gibi yaz.

Örnek 04: PHP bölümü.

```
<?php

// Pandalar inanılmaz!

?>
Pandalar harika!
```

Şimdi dosyayı tekrar çalıştır. Ne çıktısı aldık?

Örnek 05: Çıktı.

```
Pandalar harika!
```

> Durun bir dakika! Gerisi nerede?

İyi tespit, geleceğin geliştiricisi! Dosyamızdaki bir bölüm eksik. Bunun sebebi PHP etiketlerinin arasındaki her şey PHP kodu olarak görülür ve buna göre işlemden geçer.

Peki bu PHP etiketleri nelerdir? Aslında PHP açılış etiketiyle zaten tanıştın. Bu güzel arkadaşımızı hatırlıyor musun? `<?php` PHP kodunun başlangıcını bildirir. Öyleyse ne zaman sonlanır? İşte burada `?>` etiketi oyuna dahil oluyor.

Şimdi PHP etiketlerinin nasıl çalıştığını bildiğine göre, bu dosyadaki PHP kodlarını belirlemek kolay. İşte aşağıdaki satır.

Örnek 06: Açıklama.

```
// Pandalar inanılmaz!
```

Öyleyse bu satır ne yapıyor. Tam olarak hiçbir şey. Bu açıklama olarak bilinir. Bu geliştiricilerin kendi kodlarını dökümante etmelerine yardımcı olur. İlerleyen bölümlerde yorumlar hakkında daha fazla şey öğreneceğiz.

Peki, bu güzel ve kısa bir bölümdü, öyle değil mi? Şimdi biraz iyi haber zamanı. Bir sonraki bölümde ilk ***gerçek*** PHP kodunu yazacaksın.

Heyecanlandın mı? O zaman neden bekliyorsun! Sayfayı çevir.

Temel Aritmetik

Eminim programlamanın tamamen matematik olduğunu duymuşsundur, değil mi? Öyleyse matematik zamanı. Hadi başlayalım.

$$\left|\sum_{i=1}^{n} a_i b_i\right| \le \left(\sum_{i=1}^{n} a_i^2\right)^{1/2} \left(\sum_{i=1}^{n} b_i^2\right)^{1/2}$$

Şimdi X'i bulun.

Şaka yapıyorum. Aslında bu eşitlikte X yok. İşin gerçeği bu bir eşitlik bile değil, sadece kötü bir espri. Gerçek şu ki, bu karmaşanın ne yaptığına dair fikrim bile yok. Hepimiz matematik gurusu değiliz.

İfadeler

Hadi benim matematik düzeyime daha yakın birşeyler deneyelim. Nasıl PHP dosyasını oluşturacağınızı ve PHP etiketlerini nasıl açıp kapatacağınızı biliyorsunuz. Şimdi doğrudan PHP dosyasına geçelim ve onu `mat.php` olarak adlandıralım. İçeriği ise şöyle olsun.

Örnek 01: Toplama.

```
1  <?php
2
3  3 + 3;
4
5  ?>
```

Aslında, durun bir saniye. PHP kodumuzdan sonra herhangi bir çıktı vermeyeceğiz. Neden etiketi kapatma zahmetine girelim ki. Gerçek şu ki; PHP kodumuzu takip eden herhangi bir içerik yoksa çoğu PHP geliştirici bu etiketi atlar. Hadi biz de öyle yapalım.

Örnek 02: Kapatma etiketine ihtiyacımız yok

```
1  <?php
2
3  3 + 3;
```

Çok daha iyi!

Henüz matematik yeteneklerin benimkiler kadar keskin değil. Sana biraz daha yardım etmeme izin ver. Üç ile üçü topladığın zaman altı eder. Tamam, şimdi hazırsın.

`3 + 3;` satırı bir ifade içeriyor. PHP tarafından bir PHP kodu olarak değerlendirildi. Normalde PHP satırları noktalı virgül ile biter. Tıpkı şuna benziyor: `;`. İlk başta hep unutacaksınız ama endişelenmeyin. Yakında normal cümlelerinizi bile noktalı virgül ile bitireceksiniz.

Şimdi temel toplamayı öğrendiğinize göre, bu dosyayı çalıştırdığımızda çıktı ne olmalı?

> Yedi nokta beş.

Pekala, özel okuyucu. Haklı mısın görelim. İlerleyelim ve ne olacağını görmek için `php mat.php` komutunu çalışıtralım.

Örnek 03: Çıktı.

```
[burada birşey yok]
```

Vay! Kesinlikle hiçbirşey. Bu dil çok aptal. Hadi pes edelim. Pekala, yine şaka yapıyorum. Sevimsiz bir mizahım var, endişelenme alışacaksın.

Neden hiçbir çıktı alamadık? Çünkü PHP'ye herhangi bir çıktı vermesini söylemedik. PHP sadıktır. Hadi öyleyse cevabı vermesini söyleyelim. Burada `echo`'yu kullanacağız. Echo, bir ifadenin sonucunu görmek istediğimizde bize yardım eden bir PHP dil yapısıdır.

Öyleyse ifademizi `echo` içerecek şekilde değiştirelim

Örnek 04: Echo ifadesi.

```
<?php

echo 3 + 3;
```

İşte böyle. İfadeden önce `echo` kullanarak istediğimiz sonucu görebiliyoruz. Hadi uygulamamızı yeniden çalıştırmayı deneyelim. Başlıyoruz...

Örnek 05: Çıktı.

```
6
```

Vay! Altı! **YEDİ NOKTA BEŞ DEĞİL**! Bahsettiğim tam olarak buydu. İlk ifademizin PHP tarafından değerlendirilmesinin sonucunu görüyoruz. Heyecan verici, değil mi?

> Bunu hesap makinesiyle de yapabilirdim.

Biliyorum, biliyorum. Tam olarak roket bilimi değil. Roket bilimi bir sonraki başlıkta geçiy... Bir dakika, bu şakayı zaten başka bir kitapta yapmıştım. Biraz yeni malzemeye ihtiyacım var.

Aritmetik Operatörler

Bizim `3 + 3` örneğimiz basit bir kod fakat yakında çok daha büyük ve güzel şeyler öğreneceğiz. Daha fazla matematik operatörü olduğunu biliyor muydunuz. Eminim bunlar birşeyler hatırlatacaktır.

```
+ Ekleme
- Çıkarma
* Çarpma
/ Bölme
% Modül
```

Eminim daha önce bu operatörlerden bazılarını görmüştünüz. Okulda da öğrendiğiniz gibi çarpma ve bölme diğerlerinden biraz daha farklı görünüyor. Çoğu programlama dilinde bu kullanım yaygındır. Bölme işaretini klavyede yazmak görece çok daha kolaydır. Sizi endişelendirmelerine izin vermeyin, çok yakında hepsine alışmış olacaksınız.

Daha önce hiç 'Modül' operatörünü kullanmadıysanız, açıklaması çok basit. Bir bölme işleminden kalanı bulmak için kullanılabilir. Örneğin; '`3 % 2`' işleminin sonucu '`1`' olarak hesaplanabilir. Yaygın kullanımı bir sayının tek mi çift mi olduğunu belirlemek için ikiye bölme şeklindedir.

Şimdi PHP'ye düşünmesi için daha zor birşey verelim.

Örnek 06: Daha zor matematik.

```php
<?php

echo 4 + 3 * 2 / 1;
```

Sonuç nedir? Zihnimizden bu hesaplamayı yapmak zor olabilir çünkü hesap çiftlerini hangi sırayla işleyeceğimizi bilmiyoruz. Önce `4` ile `3`'ü mü toplamalıyız? Yada belki `2`'yi `1`'e bölmeliyiz. Hmm. Şaşırtmalı!

Elbette, matematikten bildiğimiz gibi eşitlikleri ayırmak için parantez kullanırız. Aynısını PHP ile de yapabiliriz. Hadi öyleyse yapalım.

Örnek 07: Ayırmak için parantez kullanımı.

```php
<?php

echo (4 + 3) * (2 / 1);
```

Şimdi `4 + 3` ve `2 / 1` daha önce değerlendirileceğini ve sonuçtaki değerlerin çarpılacağını biliyoruz. Harika, kodumuzu çalıştırıyoruz ve sonucu alıyoruz...

Örnek 08: Çıktı.

```
14
```

Harikulade, ama bu hile değil mi? Parantezler olmasaydı ne olurdu peki? Hadi parantezleri tekrar kaldırıp öğrenelim.

Örnek 09: Parantezler olmadan.

```php
<?php

echo 4 + 3 * 2 / 1;
```

Öyleyse sonuç nedir? Kodumuzu çalıştıralım.

Örnek 10: Çıktı.

```
10
```

Bu tamamiyle farklı bir sonuç. Peki neden böyle? Çünkü PHP, bizim operatörlerimizi aynı sırayla ele almıyor. Operatörlerimizin sıralamasını öğrenmek için biraz daha zaman harcayalım.

İşte PHP'nin operatörleri değerlendirme sırası.

```
* Çarpma
/ Bölme
% Modül
+ Toplama
- Çıkarma
```

En yüksek önceliğe sahip operatör tablonun en üstünde bulunmaktadır. Demektir ki PHP `4 + 3 * 2 / 1` işlemini incelerken öncelikle `3 * 2 = 6`, sonra `6 / 1 = 6` en sonunda da `4 + 6` toplamını yaparak `10` sonucuna ulaşır.

Matematiksel kod satırları yazarken herhangi bir kafa karışıklığına yol açmamak için parantezler kullanırım. Aynı zamanda kodun daha okunabilir olmasına yardımcı olduğu gibi, satırın amacını daha da belirginleştirir.

Prosedür

PHP kodu prosedürlere göre ayrıştırılır. Yani, her bir ifade tek tek okunur ve çalıştırılır. Bir satıra birden fazla ifade koymak mümkün olmasına rağmen çoğu PHP geliştiricisi arasında yaygın bir kullanım değildir. Demektir ki, kodu satır satır da ele alabiliriz. PHP dosyasmıza birden fazla ifade ekleyerek, bunu uygulamada görebiliriz. Hadi sıradakini deneyelim.

Örnek 11: Çoklu ifadeler.

```
<?php

echo 2 + 2;
echo 3 + 3;
echo 4 + 4;
echo 5 + 5;
```

Hadi dosyayı çalıştıralım..

Örnek 12: Çıktı.

```
46810
```

KIRK ALTI BİN GİGA WATT?

Sakin ol okuyucu! PHP'ye sadece sonuçların çıktılarını vermesini söyledik, çıktıya boşluk veya satırlar koymasını istemedik. Demek ki PHP değerleri doğru hesapladı. Sonuçlara boşluk eklersek...

Örnek 13: Daha belirgin çıktı.

```
4 6 8 10
```

...görüyoruz ki hesaplamalar gerçekten doğru. PHP öylesine itaatkardır ki, değerleri birbiri ardına doğrudan çıkarmıştır.

Daha önce birçok defa bahsettiğim gibi PHP çok esnek ve hoşgörülü bir dildir. Bunu test ederek görelim mi? Şimdiye kadar ifadelerimizde her bir 'kelime' (veya sayı) arasında sadece bir boşluk kullandık. Hadi biraz ilave boşlıklar koyarak aykırı bir format elde edelim, bakalım ne olacak. Değiştirilmiş kodumuz işte burada.

Örnek 14: Boşluklar.

```
<?php

echo    2          +   2;
echo                   3   +3;
echo 4+4;
echo    5+     5        ;
```

Çok hoş görünmese de, bu örnekten kodu çalıştırdığında mükemmel şekilde çalıştığını göreceksin. PHP, kendi kodu içindeki kelimelerin arasında ne kadar boşluk bırakıldığını umursamaz. Sadece onlarla başa çıkar. (Karda yürür, iz bırakmaz...)

Fark edeceksin ki bazı aritmetik işlemleri, mesela '4+4', hiç boşluk gerektirmez. Bu doğru olmakla birlikte, tüm yazım kuralları çeşitlerinde tutarlı değildir. Sıradaki örneği ele alalım.

Örnek 15: Echo'dan sonra boşluk yok.

```
<?php

echo5 + 5;
```

Bu kodu çalıştırmaya kalktığınızda PHP şöyle bir uyarı verecektir. 'Tanımlanmamış echo5 sabitinin kullanımı - 'echo5' olarak varsayıldı'. Çünkü `echo5` kelimesinin ne anlama geldiğini bilemeyecektir. Bu sebeple, her zaman kelimeler arasında birer boşluk bırakmak en iyisidir.

Eğer mazoşist isek tüm ifadeleri tek bir satıra da koyabilirdik. Burada bir örneği var.

Örnek 16: Birçok ifade, tek satır.

```
<?php echo 2 + 2; echo 3 + 3; echo 4 + 4; echo 5 + 5;
```

Bu, PHP açısından mükemmel derecede geçerlidir fakat bunu yapan çok fazla geliştirici bulamazsınız. Her bir satıra, bir tane ifade koymak kaynak dosyasını daha kolay okunur ve anlaşılabilir kılacaktır. Ayrıca aksi durum, sürüm kontrol sistemleri için de problem oluşturacaktır!

Kaynak kodunda birçok boşluk kullanmanın PHP tarafından umursanmadığını daha önce görmüştük. Aynı şekilde satır boşlukları da karakter boşluğu gibi ele alınır. Demek oluyor ki sıradaki kod öbeği tamamıyla doğrudur.

Örnek 17: Tek ifade, birçok satır.

```
<?php

echo
2
+
2
;
```

Bana inanmıyor musunuz. Gidin ve deneyin! Bu kod hedeflendiği gibi çalışmakla birlikte, en okunur kodlama şekli değildir. Sizi böyle bir kod yazarken görürsem, şaplağı yersiniz!

Oysa satır sonu boşluğunun pratik bir kullanımı da vardır. Eğer satır aşırı derecede uzun olursa, okumak bir sıkıntı haline gelir. Bunu, uygun bir okuma uzunluğuna ulaşıldığında, yeni satıra geçerek çözebiliriz. Çoğu geliştirici ayrıca yeni satıra geçtiğinde dört boşluk (veya sıradan tab) bırakır ki devam olduğu anlaşılsın. Bu, resmi yazışma metinlerinde, yeni paragrafa girintiyle başlamaya çok benzer.

Burada okunabilirlik amaçlı bir satır bölmesi görüyoruz.

Örnek 18: Satır boşluklarını temizleyin.

```
1  <?php
2
3  echo (3 * 5) / (7 / 12) * (7 * 6) + (7 % 3)
4          + (6 + 7) * (12 / 3);
```

Bu ciddi bir matematik ama umuyorum ki kolayca okuyabileceksiniz.

Ayrıca satırlar arasına boş satırlar koymak da kodunuzun daha berrak olması açısından önemlidir. Buradaki örnekte olduğu gibi.

Örnek 19: Berraklık için ilave satır boşlukları.

```
1  <?php
2
3  echo 3 + 2;
4
5  echo 7 * 7;
6
7  echo 5;
```

Gördüğünüz gibi PHP inanılmaz derecede esnek olabiliyor. Ama satırın sonuna noktalı virgül eklemeyi unutma. Çünkü seni asla affetmez.

HİÇBİR ZAMAN;

Değişkenler & Atama

Şimdi et ve patateslere geliyoruz. Değişkenler geliştirici alet çantasındaki çok kullanışlı bir o kadar da fazla suistimal edilen parçalarıdır. Haydi başlayalım, ne dersin?

Minik Kutucuklar

Değişkenleri, içerisinde bişeyler sakladığımız minik kutucukar olarak düşünmeni istiyorum. Değişkenler başında `$` işareti olan kelimelerdir. Şimdi bir örneği inceleyelim.

Örnek 01: Basit atama.

```php
<?php

$uc = 3;
```

Eger `$uc` değişkenini minik bir kutucuk olarak düşünürsek, biz içine `3` değerini koymuş olduk. Eşitlik simgesi bunu sağlıyor. Matematikte eşitlik işareti iki tarafın eşitliğini belirtiyor, fakat PHP'de bu tamamen farklı bir hikaye.

PHP'de eşitlik `=` simgesi atama operatörü olarak bilinir. Bir şeyleri ***atama*** için kullanılır. Biz PHP'ye `$uc` değişkenine `3` değerini ***atama***'sını söylüyoruz.

Eğer yukarıdaki PHP kodlarını çalıştırırsan, hiçbir çıktı vermediğini göreceksin. Çünkü bir atama işlemi, sadece bir atama işlemidir. PHP'ye herhangi bir çıktı vermesini söylemedik. Bununla birlikte, `$uc` değişkenine `3` değerini atadığımıza göre, `echo` dil oluşumunu kullanabiliriz.

Örnek 02: Bir değeri yazdırmak.

```php
<?php

// Değişkenimize üç değerini atayalım.
$uc = 3;

// Değişkenimizin değerini yazdıralım.
echo $uc;
```

İlk önce değişkenimize atama yaptık, sonra `echo` yardımıyla değişkenimizin tuttuğu değeri çıktı olarak yazdırdık. Eğer kodu çalıştırırsak, çıktı olarak `3` alırız.

Bu çok güzel, çünkü bir şeylere takma adlar verebiliriz. Bilirsin, okuldaki gıcık çocuklar gibi. Örneğin, '3.14159265359' sayısı çember sevenler arasında oldukça güzel kabul edilir, ama hatırlaması çok zordur, öyle değil mi? Haydi ona bir takma ad verelim. Ona Pete diyelim. Yok yok durun, daha iyi bir fikrim var.

Örnek 03: Düzgün bir değişken ismi.

```php
<?php

$pi = 3.14159265359;
```

İşte, `$pi` adında ve `3.14159265359` değerini tutan bir değişken yarattık. Bu kodumuzun istediğimiz yerlerinde bu değişkeni kullanarak hesaplamalar yapabileceğimiz anlamına geliyor. İşte sana birkaç örnek.

Örnek 04: Denklemler içinde değişken kullanmak.

```php
<?php

// Pi sayısını bir değişkene atayalım.
$pi = 3.14159265359;

// Çemberlerin çevresini hesaplayalım.
echo $pi * 5;
echo $pi * 3;
```

`$pi` değişkenini tanımladıktan sonra, bunu diğer tüm ifadelerimizde hesaplama için kullanabiliriz.

İstediğimiz kadar değişken yaratabilir ve bunlara istediğimiz kadar atama yapabiliriz, fakat değişken isimlerini seçerken bazı kurallara uymamız gerekiyor. Değişken isimleri sayıları, harfleri ve alt çizgiyi içerebilir. Buna rağmen, değişkenler harf veya alt çizgi ile başlamak ***zorundadır***, asla bir sayı ile başlayamazlar. Değişkenler büyük küçük harflere duyarlıdır, yani `$panda` ve `$pAnda` birbirinden farklıdır. İşte birkaç örnek.

Örnek 05: Değişkenleri isimlendirmek.

```php
<?php

$panda = 1;     // Doğru
$Panda = 1;     // Doğru
$_panda = 1;    // Doğru
$pan_da = 1;    // Doğru
$pan_d4 = 1;    // Doğru
$pan-da = 1;    // Hatalı
$4panda = 1;    // Hatalı
```

Değişler alt çizgi içerebiliyor ve büyük harfle başlayabiliyor olmasına rağmen, adlandırma için genellikle `camelCasing` (deveHörgüç notasyonu) kullanılır. Endişe etmeyin, bunun için bir deveye ihtiyaç yok.

Bu notasyondaki isimler küçük bir harfle başlar. Birden fazla kelime içeren değişkenler, devamındaki tüm kelimelerin baş harfi büyük olacak şekilde yazılır. İşte sana birkaç örnek.

Örnek 06: deveHörgüç notasyonu ile değişken isimleri.

```
<?php

$earthWormJim = 1;
$powerRangers = 1;
$spongeBobSquarePants = 1;
```

İfadelerimizden nasıl bir değer döndüğünü hatırlıyor musun? Bizim atamalarımızda aslında birer ifade. Bunun ne demek olduğunu tahmin edebiliyor musun? Evet doğru, onlarda bir değer döndürüyor. Bunu ispatlamak için emektar dostumuz `echo`'ya başvurabiliriz.

Örnek 07: Atamalardan değer döndürme.

```
<?php

echo $panda = 1337;
```

Çıktı olarak `1337` elde ettik. Bunun sebebi `$panda` değişkeninin ataması, çıktı almadan önce gerçekleşti. Bu bize zekice bir hile yapma olanağı veriyor. Bu çok kullanacağın bir özellik değil, yinede bilmek güzel. Gelin şu örneğe birlikte bakalım.

Örnek 08: Birden fazla atama.

```
<?php

$birinciPanda = $ikinciPanda = $ucuncuPanda = 1337;
```

Yukarıdaki ifade biraz delice görünebilir, ama sağdan sola doğru okursan daha mantıklı olacak. `$ucuncuPanda`'ya `1337` değeri atandı, sonra `$ikinciPanda`'ya `$ucuncuPanda`'nın değeri atandı ve sonunda `$birinciPanda`'ya `$ikinciPanda`'nın değeri atandı. Muntazam, değil mi?

Tam benim tipim

Şimdiye kadar sadece sayılarla çalışıyorduk. Sadece bu tip değerleri kullanabiliyor olsak oldukça sıkıcı olurdu, değil mi? Sanırım diğer olasılıklarıda değerlendirmemizin vakti geldi. Aşağıda PHP uygulamalarında sıklıkla kullanılan değer tipleri var.

- integer
- float
- boolean
- string

- null
- array

Bunların devamı da var, fakat işleri hemen karmaşık bir hale sokmayalım. Yavaş yavaş öğrenmemiz gerekiyor. Aşırı bilgi yüklemesi yapmak istemezsin!

Şimdi bu tiplere tek tek bakalım. İlk olarak 'integer' tipi var. Bunlar tam sayılardır, önceki uygulamalarımızda bunları kullanıyorduk.

Örnek 09: Tam sayılar.

```
1  <?php
2
3  $panda = 2;
4  $beyazPanda = -23;
```

Noktalı kayan sayılar 'float' olarak adlandırılır. Bunlar kesirli sayılar olarak da bilinir. Bunlar tam sayılara benzer şekilde kullanılabilir. Aslında bir tane kullandık bile. Arkadaşımız `$pi`'yi hatırlıyor musun? Bu bir kesirli sayıydı. Şimdi yeni bir şey yapalım, ne dersin?

Örnek 10: Kesirli sayılar.

```
1  <?php
2
3  $panda = 2.34;
4  $beyazPanda = -23.43;
```

İkili değerler 'boolean' olarak adlandırılır. Durun hemen panik olmayın. İki tabanında artimetik yapmayacağız. Bu sadece iki değerden birini alabileceğini belirtmek için kullanılır. Bir boolean sadece `true` (doğru) veya `false` (yanlış) değerini alabilir. Sonraki bölümlerde boolean değerlerin uygulamamızın akışını nasıl değiştirdiğini göreceğiz.

Örnek 11: İkili değerler.

```
1  <?php
2
3  $panda = false;
4  $beyazPanda = true;
```

Bir sonraki tipimiz dizgeler 'string' olarak adlandırılır. Bunlar bir kelime, bir karakter veya bir metin tutmak için kullanılırlar. String tipi özeldir, bu yüzden bunlar için ayrı kısa bir bölüm yazdım. Buna geri döneceğiz!

Örnek 12: String tipi.

```php
<?php

$panda = 'Normal Panda';
$beyazPanda = "Beyaz Panda";
```

Null özel bir değerdir. Hiçbir şeydir. Boş. Sıfır. Aslında sıfır değil. Sıfır nümerik bir değerdir ve bunun için integer tipini kullanabiliriz. Null tam olarak hiçbir şey demektir. Bir değişkene bir atama yapılmadan önceki değeri null'dur. Gerçekten çok işlevsel bir değerdir, ilerde bunu daha çok göreceksin.

Örnek 13: Null değeri.

```php
<?php

$pandaYok = null;
```

Diziler 'array' olarak adlandırılır ve bir başka özel tiptir. Aslına bakarsanız, bu benim diğeri arasında favorimdir. Bu yüzden bunlara tam bir bölüm ayırmaya karar verdim. Şimdilik tek bilmeniz gereken bir değerin içinde bir başka değer kümesini içerdiğidir. Vay canına! Başlangıç (Inception) filmindeki gibi mi?

Örnek 14: Diziler.

```php
<?php

$pandaSayilari = [1, 2, 3];
$dahaFazlaPandas = array(5, 6, 7, 8);
```

Gelişmiş Atama

Bir önceki bölümde değişkenler üzerinde kullanabileceğimiz operatörleri keşfettik ve atama operatöründe uzmanlaştık. Peki birden fazlasını bir arada kullansak ne olur? Yeni bir kara delik yaratıp, tüm evreni yutar mı? Kendimi çok maceraperest hissediyorum, hadi deneyelim mi?

Örnek 15: Eklemeli atama.

```
<?php

// Bir değer ver.
$panda = 3;

// Kara delik oluşturma teşebbüsünde bulun.
$panda + = 1;

// Evren dayanıyor, değişkeni döküm al.
var_dump($panda);
```

İlk önce değişkenimize tam sayı değerini atıyoruz. Sonra, eşitlik operatörünün önüne toplama operatörünü getiriyoruz ve başka bir tam sayı değeri veriyoruz.

Burada değişkeni sorguya çekmek ve sadece değerini değil hangi tipte olduğunu da öğrenmek için `var_dump()` fonksiyonunu kullanabiliyoruz (fonksiyonlarla ilgili daha fazlası yakında!).

Döküm aldığımızda ne gördük?

Örnek 16: Çıktı.

```
int(4)
```

Harika! Evren kurtuldu. Görünüşe göre elimizde dört var. Umarım bu sana mantıklı gelmiştir. `$a + $b`'nin herhangi bir atama olmaksızın değer döndürdüğünü biliyoruz, ayrıca atama operatörünün değişkenlere değer atadığını da biliyoruz. Bu ikisini de yapıyor. PHP'ye `$panda` değerini kendisi ve bir fazlası ile toplaması gerektiğini söylüyoruz.

Şimdiye kadar öğrendiğimiz operatörleri bu şekilde kullanmayı deneyebilirsin. Sadece bir sıkıntı var. Operatörleri eşitlik işaretinin diğer tarafına koyma. Bana güven, çoktan denedim. Karanlık bir yeraltı dünyasına bir kapı açıldı, yarı dinazor yarı insan yaratıklar içeri girdi ve bulunduğum şehrin altını üstüne getirdi. Sadece ev yapımı bir alev püskürtücü (gücünü PHP'den alan) sayesinde bu yaratıkları püskürtebildim. Sana da aynısı olmasından endişe ediyorum. Lütfen dikkatli ol!

Sırada, bizim bir arttırma operatörümüz var. Bunun yanında, azaltma operatörümüz olduğunu da unutmayalım. Kendisi çok daha az ilgi görüyor. Aslında, hadi gelin onun kabiliyetlerini gösterelim.

Bir örnekle devam etmeyi tercih ediyorum. Ekteki kod kesitini incele.

Örnek 17: – Sonra.

```php
<?php

// Bir değer ver.
$panda = 3;

// Değeri azalt.
$panda--;

// Değişkeni döküm al.
var_dump($panda);
```

İşte ortada, onu gördünüz mü? Güzel azaltma operatörü. Basitçe değişkenden sonra iki tane eksi işareti koyuyoruz. Ne mi yapıyor? Peki, kod kesitinin sonucu aşağıda.

Örnek 18: Çıktı.

```
int(2)
```

Gördüğün üzere `$panda` değişkeninin değeri bir azaltıldı. Bu değeri azaltmak için hızlı bir kısa yol. Benzer şekilde ++ değeri bir arttırmak için kullanılır. Dikkat et, sadece bu ikisi bu şekilde çalışır. Çarpma operatörünü de kullanmak için can atmıyor musun? Tahmin ettiğin gibi çalışmayacak!

Peki operatörü değişkenin önüne koysak ne oluyor? Haydi bakalım, olur mu?

Örnek 19: – Önce.

```php
<?php

// Bir değer ver.
$panda = 3;

// Değeri azalt.
--$panda;

// Değişkeni döküm al.
var_dump($panda);
```

Sonuç ne? Heyecanlanmadın mı?

Örnek 20: Çıktı.

```
int(2)
```

Üf, aynısı. Bu biraz sıkıcı oldu, öyle değil mi? Aslında bir sır biliyorum. Bunlar aynı değil. Tabiki çıktılar tamamen aynı gibi duruyor, ama benim örneğim bunu düzgün göstermek için yetersizdi.

Hadi şimdi farklı bir örnek yapalım. Birlikte operatörü **kullanmadan önceki** değerini göstereceğiz. Operatörü **kullandığımız andaki** değerini inceleyeceğiz. Son olarak operatörü **kullandıktan sonraki** değerini inceleyeğiz. Kullandıktan sonraki değerinin bir farkı olacağını sanmıyoruz.

Örnek 21: – operatörünün evreleri.

```php
<?php

// Bir değer ver.
$panda = 3;

// İşlem öncesinde döküm al.
var_dump($panda);

// İşlem sırasında dökün al.
var_dump(--$panda);

// İşlem sonrasında döküm al.
var_dump($panda);
```

Hadi kodu çalıştıralım. Elde ettiğimiz üç değer ne?

Örnek 22: Çıktı.

```
int(3)
int(2)
int(2)
```

İlk değer üç. Bunu bekliyor olmalıydık, yani demek istediğim sadece atamayı yaptık, değil mi? Azaltma operatörünün kullanıldığı satırdaki değer iki. Sonuçtaki değer de iki. Demek ki azaltma ikinci satırda yapılmış.

Şimdi operatörü değişkenin diğer tarafına alalım, olur mu? Şunun gibi:

Örnel 23: – operatörünün evreleri, bölüm 2.

```
<?php

// Bir değer ver.
$panda = 3;

// İşlem öncesinde döküm al.
var_dump($panda);

// İşlem sırasında dökün al.
var_dump($panda--);

// İşlem sonrasında döküm al.
var_dump($panda);
```

Aradaki farkı anlamak için dikkatli bak. Şimdi sonuçlara tekrar göz atalım.

Örnek 24: Çıktı.

```
int(3)
int(3)
int(2)
```

Hey!? Ortadaki değer farklı! Neden azalma olmadı? Şöyle, operatörün yerini değiştirerek, PHP'ye değişkenin değerini mevcut satırdan **SONRA** değiştirmesini söyledik. Sonuçta elde ettiğimiz değer tabiki aynı oluyor.

Şimdi özetleyeyim.

```
$deger-- - Değeri bu satırdan *sonra* değiştir.
--$deger - Değeri bu satırda değiştir.
```

Bu neden kullanışlı? Şöyle, size bir kullanım alanı göstereyim. Eminim biraz yaratıcıysan daha fazlasını bulabilirsin. Bir **sonraki** satırda değişikliği yapan operatörü kullanarak, bir başka değişkeni mevcut değere atayabilir ve orjinal değeri bir azaltabiliriz. Şunun gibi:

Örnek 25: Ata ve arttır.

```php
<?php

// Bir değer ver.
$panda = 3;

// Atama yap, sonra arttır.
$pandaFriend = $panda++;
```

Burda ne yaptık, bir satır kazandık. Bir nevi kısayol. Eğer arttırma operatörünü kullanmakdan yapsaydık şu şekilde olurdu.

Örnek 26: Attırma açıklandı.

```php
<?php

// Değer ver.
$panda = 3;

// Ata.
$pandaFriend = $panda;

// Arttır.
$panda = $panda + 1;
```

Döngüleri ele aldığımız bölümde, bunun daha farklı kullanım alanlarınıda bulacaksın. Bir sonraki bölümde ‘string’ tipine yakından bakacağız.