HTML5, CSS3 ve JavaScript ile Web Tasarimi

Burak Tokak

İçindekiler

5

CSS ile Sayfa Düzeni Oluşturmak (Layout)

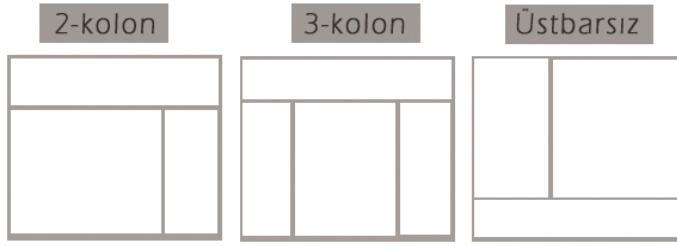
Bu Bölümde

Önceki bölümümüzde, CSS özelliklerinin nasıl kullanılacağını ve işimize yarayacak bazı özelliklerin işlevlerini öğrendik.

Bu bölümde, öğrendiğimiz işlevleri kullanarak CSS sayesinde farklı sayfa düzenlerini (layout) en verimli yoldan nasıl oluşturabileceğimizi inceleyeceğiz.

GİRİŞ

İnternet sitelerinin en büyük amacı, içeriğini kullanıcıya sunmaktır. Birçok internet sitesi, kullanıcı deneyimini arttırmak için içeriklerini birden fazla kolonda (bölümde) gösterirler. Bu kolonlar sistemine **Sayfa İskeleti** ya da **Layout** diyoruz. Layoutlar genellikle sahip oldukları kolon sayısına göre tehşis edilir. Magazin ve haber siteleri gibi kullanıcının ilgisini farklı noktalara çekmek isteyen internet siteleri 3 kolonlu sayfa düzenleri kullanırken, blog ya da tek tip içerik bulunduran internet siteleri (video siteleri, resim galerileri vs.) 2 kolonlu sayfa düzeni kullanır. Bir ürünü veya şirketi tanıtım amaçlı yapılmış internet siteleri, teknik bilgi ya da kullanım kılavuzu makaleleri bulunan internet siteleri vb. ise genellikle tek kolonlu layoutlar kullanır.



Görselde çok kullanılan 3 farklı layout tipini görüyoruz.

2-kolon: Sayfa düzeninde genellikle bir üst bölüm bulunurken, içeriğin kendisi geniş olan kolonda bulunur, genellikle tek tip içerik bulunduran internet sitelerinde kullanılan bu tür sayfa düzenleri içindir. Sağ bölüm ise; kullanıcının ilgisini çekebilecek diğer içerikler bulunur.

3-kolon: Bu sayfa düzeninde, sunulacak çok fazla farklı içerik bulunan internet siteleri için uygundur. Yan bölümlerde diğer içerik sayfaları bulunurken, 2-kolonlu sayfa düzeninde olduğu gibi, asıl içerik **geniş** (orta) bölümde bulunur.

KLASİK LAYOUT ÖRNEKLERİ

Sayfa iskeletleri genellikle CSS ile oluşturulmaktadır. HTML içerisindeki **Table** elementi kullanarak da layout oluşturabilirsiniz, fakat table elementi ile her tarayıcıda aynı çıktıyı almak biraz zordur. Ayrıca Table elementi, CSS kadar güçlü bir altyapıya da sahip olmadığından CSS'in kullanılması şiddetle tavsiye edilir.

NOT Bunun yanında fazla karşılaşılmamasına rağmen, table ve kutu modelini table gibi display ederek sayfa iskeletinde verimli bir görüntü elde edebiliyoruz.

2-Kolonlu Layout Örneği

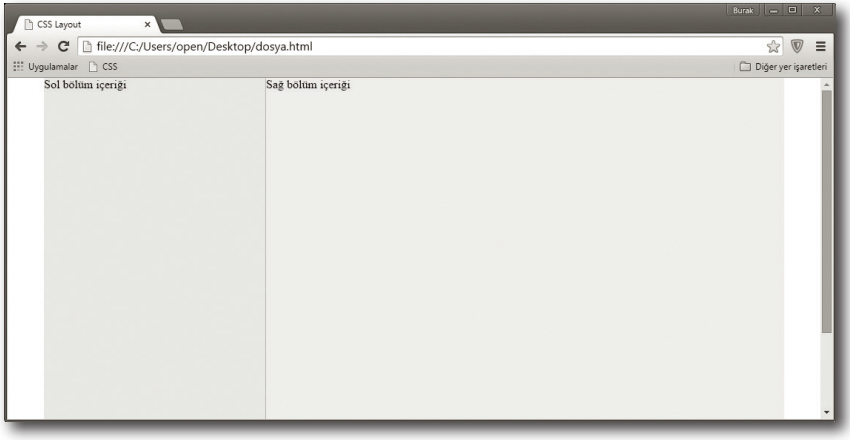
HTML Kodu

```
<div class="tasiyici">
  <div class="solbolum">Sol bölüm içeriği</div>
  <div class="sagbolum">Sağ bölüm içeriği</div>
</div>
```

CSS Kodu (style.css)

```
* { padding:0px; margin:0px;}
.tasiyici { width:1000px; margin:auto; }

.solbolum { float:left; width:299px; height:600px; background:#f1f1f1;
border-right:1px solid #ccc;}
.sagbolum { float:right; width:700px; height:600px; background:#f5f5f5; }
```



DİKKAT Örnekte gördüğünüz gibi **1000px** genişliği, hizaladığım div'lere paylaştırdım. Burada dikkat etmemiz gereken konu, **border** genişliği olan **1px**'in div içerisinde sayılmıyor olması, yani eğer **300px** ve **700px** olarak div genişliklerimi ayarlasaydım, toplamda genişlik **1001px** olacağından, **sagbolum** div'i bir alt satıra kayacaktı. Bu yüzden **solbolum** div'i için **300** yerine **299px** genişlik atadık.

Üst ve Alt Bölümlü Layout Örneği

HTML Kodu

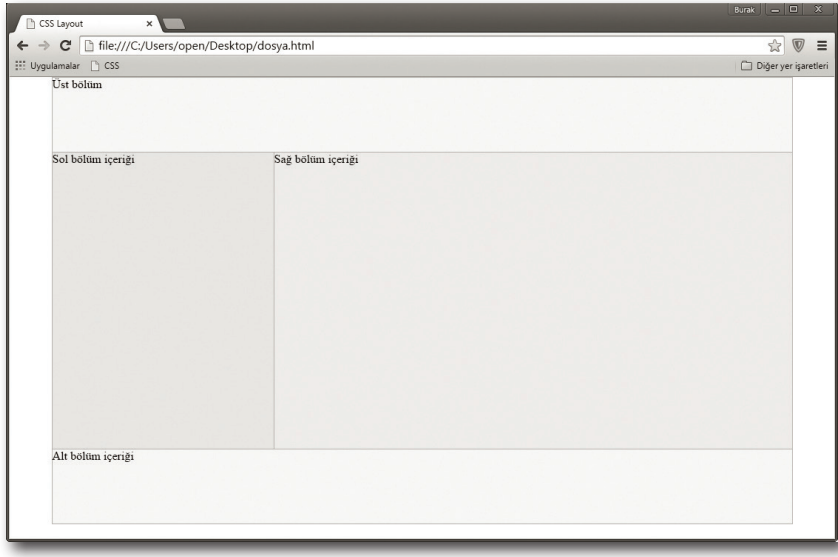
```
<div class="tasiyici">
  <div class="ustbolum">Üst bölüm</div>
  <div class="solbolum">Sol bölüm içeriği</div>
  <div class="sagbolum">Sağ bölüm içeriği</div>
  <div class="clear"></div>
  <div class="altbolum">Alt bölüm içeriği</div>
</div>
```

CSS Kodu (style.css)

```
* {padding:0px; margin:0px;}
.tasiyici {width:1000px; margin:auto; border:1px solid #ccc;}

.ustbolum {height:100px; width:1000px; border-bottom:1px solid #ccc; background:#fafafa;}
.solbolum {float:left; width:299px; height:400px; background:#f1f1f1; border-right:1px solid #ccc;}
.sagbolum {float:right; width:700px; height:400px; background:#f5f5f5;}
.altbolum {height:100px; border-top:1px solid #ccc; background:#fafafa;}

.clear {clear:both;}
```



Bu örneğimizde, bir önceki örneğe ek olarak, üst ve alt bölümler ekledik. `.ustbolum` class'lı div için **100px** yükseklik verdik ve **1000px** genişlik verdik, gerekli border'ları verdikten sonra böyle bir görüntüye ulaştık. `.clear` class'ı ile kullandığımız temizleme divi, `solbolum` ve `sagbolum` div'lerinin ikisinin de `float` değerine sahip olmasından dolayı, sayfanın devam noktası bu div'lerin başlangıçları olacaktı. Bu yüzden altbolum div'i, bu iki div'in başladığı yerde ve altlarında yer alıyor olacaktı. `sagbolum` div'i sonrası `both clear` özelliğine sahip `clear` div'i ile bunu engelledik.

NOT

Dikkatinizden kaçmadıysa, `ustbolum` ve `altbolum` div'leri aynı genişlikte sayfaya yansıdı. Fakat `altbolum` div'inde, `ustbolum` div'ine karşın **1000px width** özelliği bulunmuyor. Div elementi default olarak **block display** özelliğine sahip bir element olduğundan, genişliği yüzde yüz olarak doldurur. Yani bu durumda div için taşıyıcının boyutunu alır.

3-Kolonlu Layout Örneği

HTML Kodu

```
<div class="tasiyici">
  <div class="ustbolum">Üst bölüm</div>
  <div class="solbolum">Sol bölüm içeriği</div>
  <div class="ortabolum">Orta bölüm içeriği</div>
  <div class="sagbolum">Sağ bölüm içeriği</div>
  <div class="clear"></div>
  <div class="altbolum">Alt bölüm içeriği</div>
</div>
```

CSS Kodu (style.css)

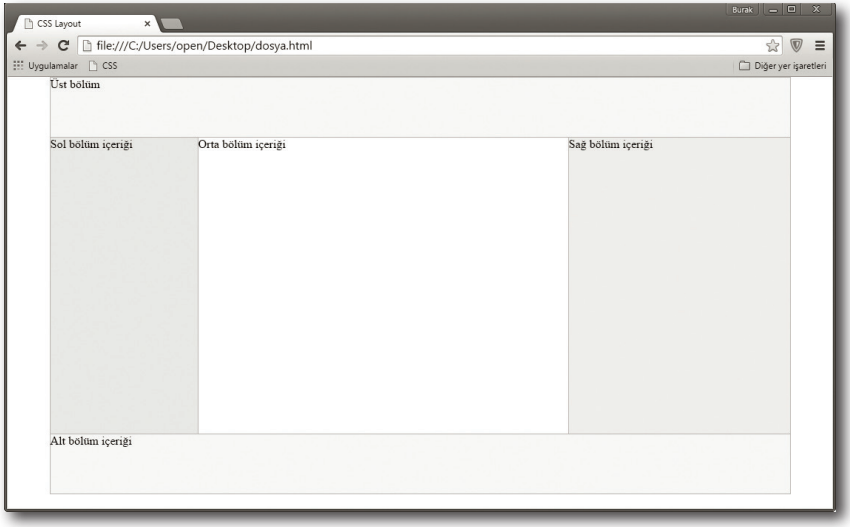
```
* {padding:0px; margin:0px;}
.tasiyici {width:1000px; margin:auto; border:1px solid #ccc;}

.ustbolum {height:80px; border-bottom:1px solid #ccc; background:#fafafa;}

.solbolum {float:left; width:199px; height:400px; background:#f1f1f1;
border-right:1px solid #ccc;}
.ortabolum {float:left; width:500px; height:400px; background:#fefefe;}

.sagbolum {float:right; width:299px; height:400px; background:#f5f5f5;
border-left:1px solid #ccc;}
.altbolum {height:80px; border-top:1px solid #ccc; background:#fafafa;}

.clear {clear:both;}
```



Yeni örneğimizde 3 kolonlu bir layout oluşturduk. Önceki örneğe ek olarak bir `ortabolum` div'i oluşturduk ve bu div için de `left float` özelliği verdik. Sol ve Sağ bölümlerin genişliklerini ayarlayıp, `ortabolum` div'i için **500px** genişlik atadık. Kenarlıkları ve kenarlıklardan dolayı çıkartmamız gereken pixel'leri de çıkarttıktan sonra böyle bir görüntü oluştu.

NOT

Orta bölüm için neden `left float` özelliği kullandığımızı şöyle özetleyebiliriz, eğer kullanmasaydık, div elementi block olarak davranıp, sağ bölümü aşağıya atacaktı. Right özelliği kullansaydık, `right float` için sayfada önde olan elementin önceliği daha fazla olduğundan orta bölüm Sağa geçecekti.

Bunun sonrasında, kutu modelini kullanarak, farklı kombinasyonlar ile layout kompozisyonları oluşturabilirsiniz. Bahsettiğimiz örnekler bu layoutların en basit ve en çok kullanılanlarıdır.

Sabitlenmiş Bölümlü Layout Örnekleri

Klasik layoutlara ek olarak bahsetmek istediğim son zamanalarda işlevselliği nedeni ile popüler olan diğer bir layout tipi bir bölümü sabitlenmiş hizalamalar.c Çoğu zaman üst bölümü sabitlenmiş, bazen de özel işlevsel amaçlarla yan bölümleri sabitlenmiş sayfa hizalamaları internette en çok kullanılan layout tipleridir.

10

Javascript Koşul Kontrolleri ve Döngüler

Bu Bölümde

Önceki bölümümüzde Javascript betik dilinin genel özelliklerini ve syntax'ını öğrendik. Javascript üzerindeki veri türleri ve değişkenlerin mantığının nasıl çalıştığını anlattık.

Bu bölümde ise, aynı şekilde Javascript'in mantıksal işlevleri yerine getirebilmesi ve birden fazla, belirli bir indis üzerinde gerçekleşen işlemlerin daha hızlı halledilmesi için kullandığımız koşul kontrolleri ve döngülerden bahsedeceğiz.

Giriş

Javascript'in C furyası programlama dillerinin syntax'ına benzer bir söz dizilimi olduğundan bahsetmiştik. Eminim kitabı okuyanlar arasında C dilleri ile ilgilenmiş olanlar bulunuyordur, aslına bakılırsa bu bölümde anlatacağımız konu, çoğu programlama dillerinde kullanılan ve çok benzer bir söz dizilimi ile oluşturulan bir konudur.

Bir programlama dilinin temelini oluşturan koşul kontrolleri, kodunuz içerisinde belirli bir durum olduğunda çalışmasını istediğiniz veya istemediğiniz kodların yönetimini sağlarken; döngüler, kodunuz içerisinde belirli değişkenlerin değişmesi ile kodun birkaç kez çalışmasını istediğiniz durumlarda işimize yarar.

JAVASCRIPT KOŞUL KONTROLLERİ (IF...ELSE)

Örneğin; online patates satan bir uygulama geliştiriyorsunuz. Bu uygulamayı oluştururken karşınıza çıkan mikro bir problem, kullanıcıların 3 kilo altında patates almasını istememeniz. Bu durumda kullanıcıya kaç kilo patates aldığını sorduğunuzda 3 kilodan az bir miktar verdiğinde bu kadar az patates alamayacağını söylemeniz gerekiyor, 3 kilodan fazla isterse uygulamanız işlemlerine devam edecek. Bu gibi işlemleri `if…else` koşullandırma blokları ile gerçekleştiriyoruz.

JS Kodu

```
if(koşul){
    // Koşul sağlandığında gerçekleştirilecek işlemler
}
if(koşul){
    // Koşul sağlandığında gerçekleştirilecek işlemler
}else{
    // Koşul sağlanmazsa gerçekleştirilecek işlemler.
}
if(koşul1){
    // 1. Koşul sağlanırsa
}else if(koşul2){
    // 2. Koşul sağlanırsa
}else{
    // Her iki koşul da sağlanmazsa
}
```

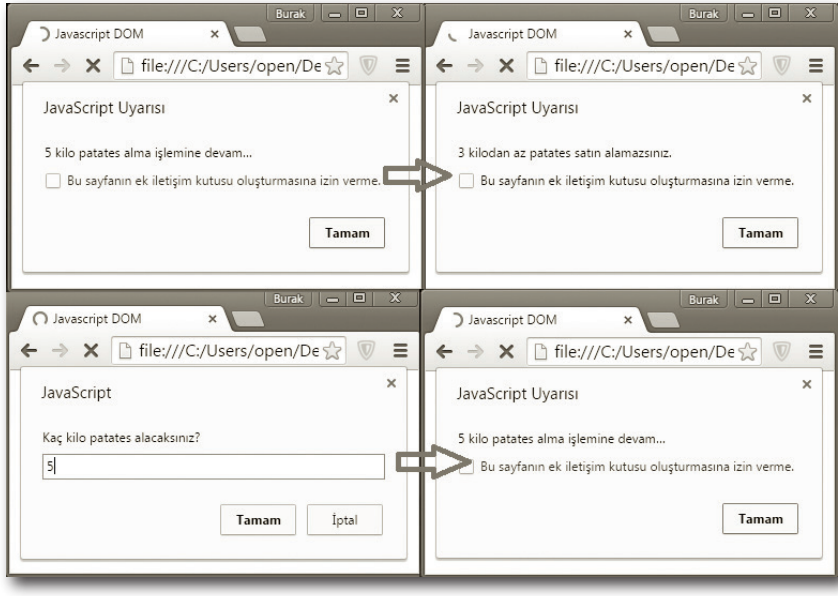
NOT Burada koşul ile belirttiğimiz durum bir değişkenin bir sayıya eşit, eşit değil, büyük veya küçük gibi mantıksal doğrulamlardır. Bir koşulun değeri boolean (`true/false`) olarak değerlendirilir. Çıkan değerin doğru ya da yanlış olmasına göre **if statement**'ı programı yönlendirir.

İsterseniz bahsettiğimiz patates satış uygulamasını bir örnek ile yazmaya çalışalım;

JS Kodu

```
var kacKiloPatates = prompt("Kaç kilo patates alacaksınız?");

if(kacKiloPatates < 3){
   alert("3 kilodan az patates satın alamazsınız.");
}else{
   alert(kacKiloPatates + " kilo patates alma işlemine devam…");
}
```



Örneğimizde 3'den küçük ve büyük değerler verildiğinde if ifadesinin nasıl davrandığını inceleyebilirsiniz.

Değişkenin Varlığı Kontrolü

If ifadesi ile bir değişkenin varlığını veya doğruluğunu kontrol edebilirsiniz.

JS Kodu

```
var degisken1 = "Dolu";
var degisken2;

if(degisken1){
   alert("degisken1 atanmış.");
}else{
   alert("degisken1 henüz atanmamış.");
}

if(degisken2){
   alert("degisken2 atanmış.");
}else{
   alert("degisken2 henüz atanmamış.");
}
// Çıktı: (1.Uyarı) degisken1 atanmış. (2.Uyarı) degisken2 henüz atanmamış.
```

Örneğimizde atanmış dolu bir değişken ile atanmamış boş bir değişkeni if koşul belirteci içerisinde kontrol ettirerek sonucu yorumlattık.

NOT

If ifadesi içerisine yazılan çıktı değişkeni (koşulun sonucu) boolean'a cast edilip değerinin true/false olarak yorumlandığını düşünebilirsiniz. Buna göre **dolu** string'i cast edildiğinde true değeri dönecek, undefined bir değişken cast edildiğinde false dönecektir.

KARŞILAŞTIRMA (COMPARISON) OPERATÖRLERİ

Karşılaştırma operatörleri; değişkenler ile bir koşul oluşturmamızı sağlayan koşul operatörlerdir. Bu operatörler ile oluşturduğumuz koşullar doğru ya da yanlış olarak sonuç verir.

Eşittir (==) ve Eşit Değildir (!=) Operatörleri

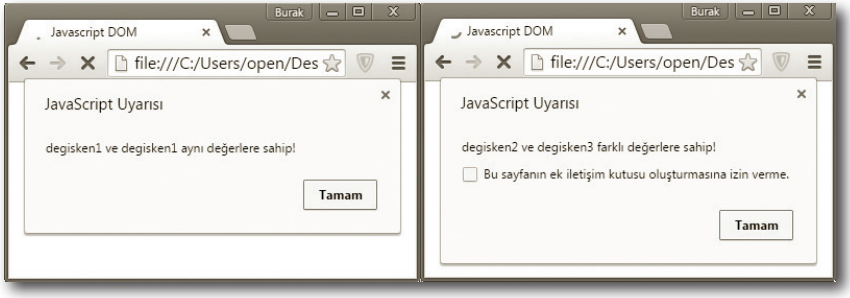
İki değerin birbirine eşit olup olmadığını kontrol eden operatörlerdir.

JS Kodu

```
var degisken1 = 10;
var degisken2 = 10;
var degisken3 = 15;

if(degisken1 == degisken2){
   alert("degisken1 ve degisken1 aynı değerlere sahip!");
}else{
   alert("degisken1 ve degisken1 farklı değerlere sahip!");
}

if(degisken2 != degisken3){
   alert("degisken2 ve degisken3 farklı değerlere sahip!");
}else{
   alert("degisken2 ve degisken3 aynı değerlere sahip!");
}
```



Örneğimizde 3 farklı değişken ile `if` koşulu içerisinde kullandığımız == ve != operatörlerinin kullanımlarını ve davranışlarını görebilirsiniz. Burada eşit değildir operatörü, değil olması nedeni ile olumsuz gibi duruyor olmasına rağmen `degisken2` ve `degisken3` birbirine eşit olmadığından dolayı doğru değeri dönecektir.

DİKKAT

Eşittir koşul operatörünün 2 adet eşittir işaretinden oluştuğuna dikkat ediniz. İki değeri birbirine eşitlemek için kullandığımız tek işaretli eşittir operatöründen tamamen farklıdır. Javascript'te yapabileceğiniz en büyük syntax hatalarından birisi dalgınlık ile çift eşittir yerine kontrol sırasında tek eşittir kullanmaktır.