

# HTML5, CSS3 VE JAVASCRIPT İLE WEB TASARIMI

---

BURAK TOKAK

# İÇİNDEKİLER

<b>BÖLÜM 1: WEB GELİŞTİRİCİLİĞİNE GİRİŞ</b>	<b>1</b>
Kavramlar	2
Web Teknolojileri	2
Client-Side Web Teknolojileri	2
Server-Side Web Teknolojileri	4
Web Geliştiricisi (Web Developer)	5
Freelancer (Bağımsız) Web Geliştiricisi	6
Syntax (Sözdizimi)	7
HTTP	7
FTP	7
Domain (Alan Adı) ve Hosting (Yer Sağlayıcı) Kavramları	7
Neler Öğrendik?	8
<b>BÖLÜM 2: HTML'E BAŞLARKEN</b>	<b>9</b>
Giriş	10
HTML'in Çalışma Mekanizması	11
Biçimlendirme Dilleri	11
İnternet Tarayıcıları	12
HTML Kodlaması için Ortam ve Program Gereklilikleri	13
HTML Etiketleri ve Elementleri	14
HTML Attribute Tanımı	15
HTML Dosyasının Genel Yapısı	16
HTML, Head ve Body Hiyerarşisi	18
HTML Özel Karakterlerinin Kullanımı	19
HTML, HEX ve RGB Renk Düzenleri	21
Neler Öğrendik?	23

**BÖLÜM 3: HTML ELEMENTLERİ VE KULLANIMLARI 25**

Giriş	26
Head İçerisinde Kullanılan Elementler	26
Title Elementi	26
Style Elementi	27
Link Elementi	27
Base Elementi	28
Meta Elementi	29
Body İçerisinde Kullanılan Elementler	32
Yazı Şekillendirme Elementleri	32
Hizalama ve Yerleştirme Elementleri	36
Fazla Kullanılan Teknik Elementler	43
Bağlantı ve Resim Ekleme Elementler	44
Listeleme Elementleri	47
Form ve Input Elementleri	50
Neler Öğrendik?	54

**BÖLÜM 4: CSS İLE BİÇİMLENDİRME 55**

Giriş	56
Neden CSS Kullanıyoruz?	56
CSS Çalışma Mekanizması ve Syntax'ı	57
Syntax (Sözdizimi)	58
Seçiciler (Selectors)	59
CSS Özellikleri (Property) ve İşlevleri	65
Yazı Biçimlendirme Özellikleri	65
Kutu Biçimlendirme Özellikleri	70
Kutu Hizalama Özellikleri	78
Position Özelliği ile Konumlandırma	85
Neler Öğrendik?	88

**BÖLÜM 5: CSS İLE SAYFA DÜZENİ OLUŞTURMAK (LAYOUT) 89**

Giriş	90
Klasik Layout Örnekleri	90
2-Kolonlu Layout Örneği	91
Üst ve Alt Bölümlü Layout Örneği	92
3-Kolonlu Layout Örneği	93
Sabitlenmiş Bölümlü Layout Örnekleri	94
Üst Bölümü Sabitlenmiş Layout Örneği	95
Yan Bölümü Sabitlenmiş Layout Örneği	97
Grid Bölümlü Layout Örneği	98
Neler Öğrendik?	101

**BÖLÜM 6: CSS PSEUDO-CLASS VE TRANSITION 103**

Giriş	104
Pseudo-Classlar ve Kullanımları	104
First-Child ve Last-Child Ara-Seçicileri ve Kullanımı	104
Not Ara-Seçicisi ve Kullanımı	105
Hover Ara-Seçicisi ve Kullanımı	106
Focus Ara-Seçicisi ve Kullanımı	109
Active Ara-Seçicisi ve Kullanımı	110
Nth-Child Ara-Seçicisi ve Kullanımı	112
Transition ve Kullanımı	113
Neler Öğrendik?	119

<b>BÖLÜM 7: CSS TRANSFORM VE FONKSİYONLARI</b>	<b>121</b>
GİRİŞ	122
Transform Fonksiyonları ve Kullanımları	123
translate() Fonksiyonu	123
scale() Fonksiyonu	124
rotate() Fonksiyonu	126
skew() Fonksiyonu	128
Transform Fonksiyonları ile Transition	129
Neler Öğrendik?	132
<b>BÖLÜM 8: CSS ANİMASYONLAR (ANIMATION)</b>	<b>133</b>
Giriş	134
Diğer (Yardımcı) Animation Özellikleri	135
CSS3 Animasyon Örnekleri	137
Neler Öğrendik?	140
<b>BÖLÜM 9: JAVASCRIPT'E BAŞLARKEN</b>	<b>141</b>
Giriş	142
Javascript Ekosistemi ve Diğer Programlama Dilleri	142
Javascript Syntax'ı ve Client Uygulamaları	143
HTML Sayfalarına Javascript Yerleştirmek	143
Javascript İçerisinde Yorum Satırları	144
Javascript Açılır Pencereleeri	145
Javascript Değişkenleri (Variables)	149
Javascript Öntanımlı Değişken Fonksiyonları	154
Neler Öğrendik?	156

**BÖLÜM 10: JAVASCRIPT KOŞUL KONTROLLERİ VE DÖNGÜLER 157**

Giriş	158
Javascript Koşul Kontrolleri (if...else)	158
Karşılaştırma (Comparison) Operatörleri	160
Mantıksal (Logical) Operatörleri	164
Javascript Döngüleri (for - while)	167
For Döngü İfadesi ve Kullanımı	168
While Döngü İfadesi ve Kullanımı	169
Neler Öğrendik?	171

**BÖLÜM 11: JAVASCRIPT FONKSİYONLARI 173**

Giriş	174
Javascript'te Fonksiyon Tanımı ve Özellikleri	174
Javascript Fonksiyonlarının Özellikleri	177
Neler Öğrendik?	179

**BÖLÜM 12: JAVASCRIPT OBJELER VE DOM 181**

Giriş	182
Javascript Obje Veri Tipi ve Özellikleri	182
Javascript Document Object Model (DOM)	184
Element Objelerini Seçmek	185
Element Objelerinin İçeriğini Değiştirmek	186
Element Objelerinin Stillerini Değiştirmek	188
Element Objelerinin Nitelik Tanımlayıcılarını Değiştirmek	189
Document Objesinin Diğer Elemanları ve Kullanımı	190
Neler Öğrendik?	192

<b>BÖLÜM 13: HTML OLAY TANIMLAYICILARI</b>	<b>193</b>
Giriş	194
HTML Olay Tanımlayıcılarının İşlevleri	194
onClick Olay Tanımlayıcısı	195
onMouseOver ve onMouseOut Olay Tanımlayıcıları	197
Klavye ile Alakalı Olay Tanımlayıcıları	199
onLoad Olay Tanımlayıcısı	201
addEventListener Yöntemi ile Olay Tanımlama	202
Neler Öğrendik?	204
<b>BÖLÜM 14: CSS MEDIA SORGUSU İLE RESPONSIVE TASARIM</b>	<b>205</b>
Giriş	206
CSS ile Sayfa Düzenini Ölçekleme	207
Ölçekler ile calc() Fonksiyonu Kullanımı	208
Aygıtlar Üzerinde Ölçekleme	213
@media Kuralı ile Ekran Boyutlarını Yönetmek	214
Farklı Responsive Yöntemleri	217
Neler Öğrendik?	217
<b>BÖLÜM 15: JAVASCRIPT İLE UYGULAMALI ÖRNEKLER</b>	<b>219</b>
Giriş	220
Javascript ile Menü Açma/Kapatma İşlemleri	220
Javascript ile Basit Web Oyunları	225
Farklı Renkte Kutuya Tıklama Oyunu	225
Neler Öğrendik?	230
İndeks	231

# 5

## CSS İLE SAYFA DÜZENİ OLUŞTURMAK (LAYOUT)

### BU BÖLÜMDE

Giriş	90
Klasik Layout Örnekleri	90
Sabitlenmiş Bölümlü Layout Örnekleri	94
Neler Öğrendik?	101

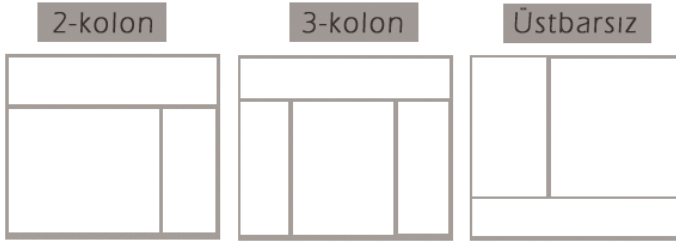
Önceki bölümümüzde, CSS özelliklerinin nasıl kullanılacağını ve işimize yarayacak bazı özelliklerin işlevlerini öğrendik.

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



## Giriş

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** diyorumuz. Layoutlar genellikle sahip oldukları kolon sayısına göre teşhis 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 tarafta 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 ÖRNEĞİ

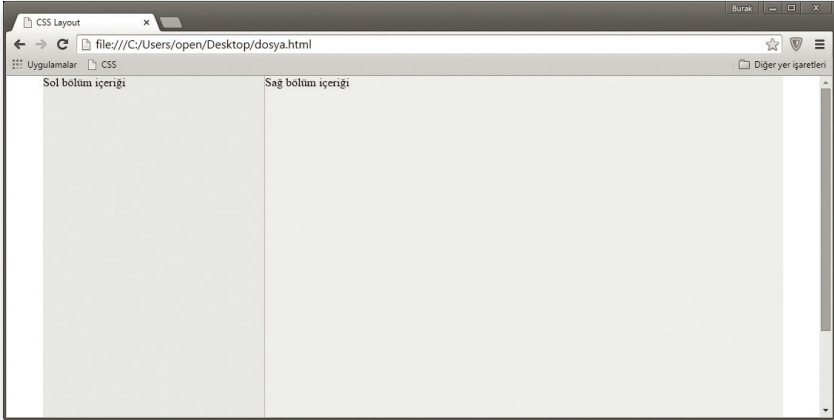
### HTML Kodu

```
<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 gördüğümüz 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 ÖRNEĞİ

### HTML Kodu

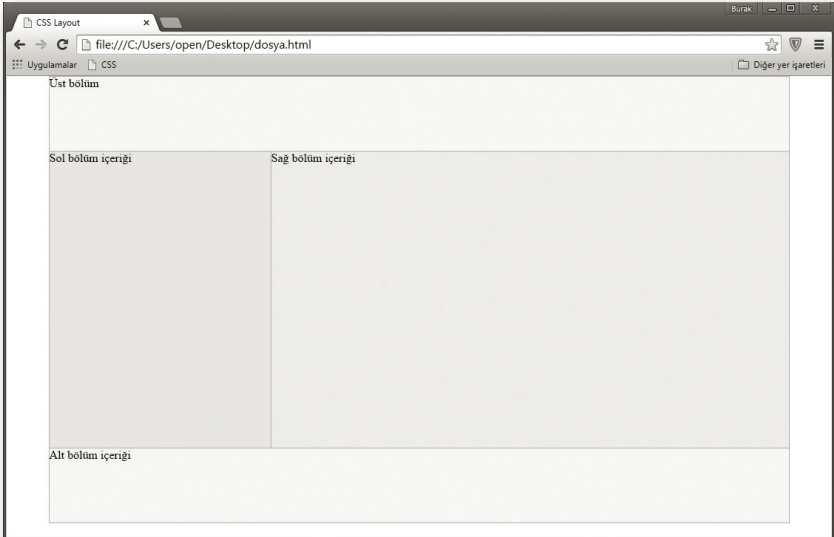
```
<div class="tasiyici">
  <div class="ustbolum">Üst bölüm</div>
  <div class="solbolum">Sol bölüm içeriği</div>
  <div class="sagbolum">Sağ bölüm içeriği</div>
  <div class="clear"></div>
  <div class="altbolum">Alt bölüm içeriği</div>
</div>
```

### 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 ÖRNEĞİ

#### HTML Kodu

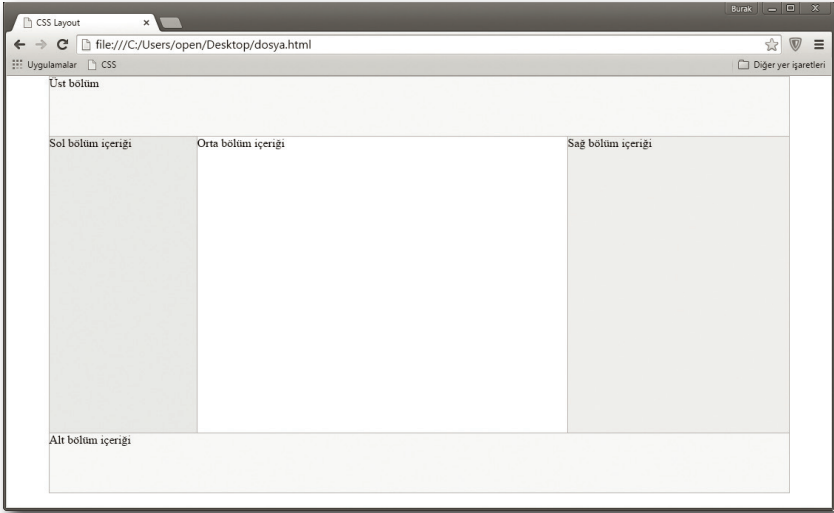
```
<div class="tasiyici">
  <div class="ustbolum">Üst bölüm</div>
  <div class="solbolum">Sol bölüm içeriği</div>
  <div class="ortabolum">Orta bölüm içeriği</div>
  <div class="sagbolum">Sağ bölüm içeriği</div>
  <div class="clear"></div>
  <div class="altbolum">Alt bölüm içeriği</div>
</div>
```

#### 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 ortabölüm div'i oluşturduk ve bu div için de `left float` özelliği verdik. Sol ve Sağ bölümlerin genişliklerini ayarlayıp, ortabölüm 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.

## SABİTLENMİŞ BÖLÜMLÜ LAYOUT ÖRNEKLERİ

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

# 10

## JAVASCRIPT KOŞUL KONTROLLERİ VE DÖNGÜLER

### BU BÖLÜMDE

Giriş	158
Javascript Koşul Kontrolleri (if...else)	158
Javascript Döngüleri (for - while)	167
Neler Öğrendik?	171

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 koşullu 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
}
```

---

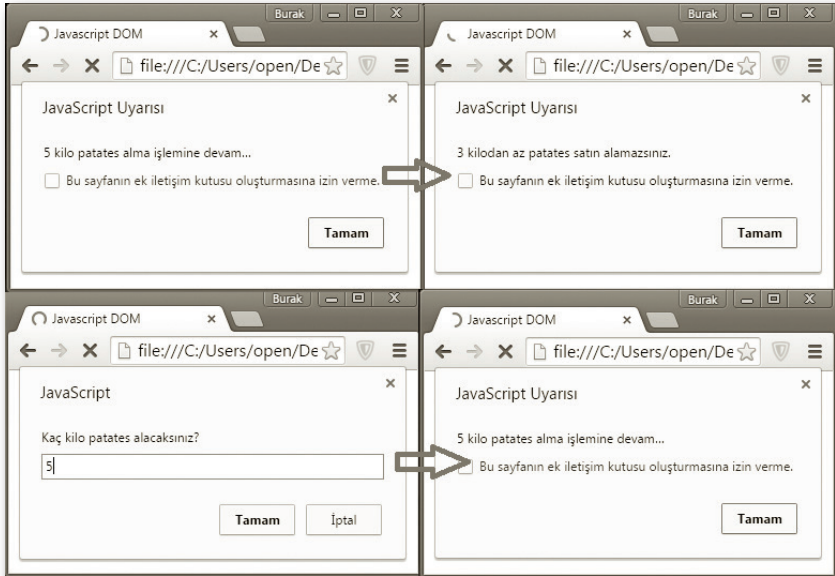
**NOT**

Burada koşul ile belirttiğimiz durum bir değişkenin bir sayıya eşit, eşit değil, büyük veya küçük gibi mantıksal doğrulamalardır. Bir koşulun değeri boolean (true/false) olarak değerlendirilir. Çıkan değer doğru ya da yanlış olmasına göre **if statement**'i 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 değer birbirine eşit olup olmadığını kontrol eden operatörlerdir.

## JS Kodu

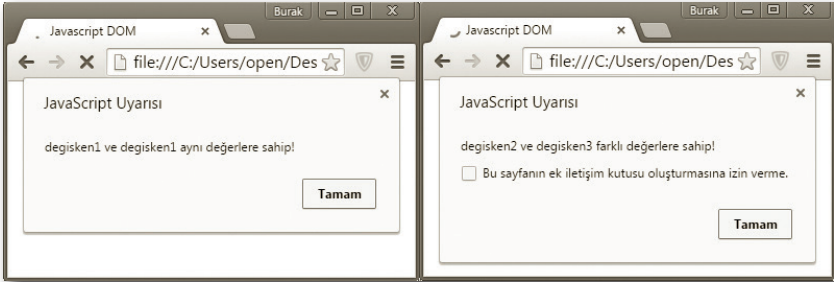
```

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.