

WEB & GRAFİK

Web Tasarımcıları İçin CSS

Onur Öztaşkıran

Editör Koray Al

39-8

Web Tasarımcıları İçin CSS

Onur Öztaşkıran

Editör: **Koray Al**

Kapak Tasarımı: **Melih Sancar**

Grafik Uygulama: **Tuna Erkan**

Yayın Yönetmeni: **Selçuk Tüzel**

Genel Yayın Yönetmeni: **Mehmet Çomlekçi**

1. Basım: Eylül 2008

Bilge Adam Yayınları: 41

Eğitim Yayınları Dizisi: 41

ISBN: 978-605-5987-39-8

Copyright © 2008, Bilge Adam Bilgisayar ve Eğitim Hizmetleri San. ve Tic. A.Ş.

Eserin tüm yayın hakları Bilge Adam Bilgisayar ve Eğitim Hizmetleri San. ve Tic. A.Ş.'ye aittir. Yayınevinden yazılı izin alınmadan kısmen ya da tamamen alıntı yapılamaz, hiçbir şekilde kopya edilemez, çoğaltılamaz ve tekrar yayımlanamaz. Bilge Adam'ın öğrencilerine ücretsiz armağanıdır, para ile satılamaz.

Bilge Adam Bilgisayar ve Eğitim Hizmetleri San. ve Tic. A.Ş.

19 Mayıs Mahallesi, 19 Mayıs Caddesi, UBM Plaza, No: 59-61, Kat: 4-7; Şişli, İstanbul

Telefon: (212) 272 76 00 – (212) 217 05 55 Faks: (212) 272 76 01

www.bilgeadam.com - info@bilgeadam.com

Tanıtım nüshasıdır, para ile satılamaz.

Sunuş

İnternet ve web sitelerinin hem görsel hem de teknik olarak daha da zenginleştiği bu yıllarda, artık oldukça bilinçlenmiş olan İnternet kullanıcılarına eski tekniklerle bir şeyler sunma devrinin sonuna gelmiş bulunmaktayız.

Birçok İnternet kullanıcısı, beğendiği web sitelerini RSS (Really Simple Syndication) beslemeyle-riyle takip ediyor, bilgiye daha hızlı ulaşma yollarını arıyor ve daha hızlı açılan, daha esnek, daha kolay ulaşılabilen, kendi tercihleri doğrultusunda daha fazla özelleştirebileceği ve daha fazla platformdan erişebileceği web sitelerini tercih ediyor.

Tasarımcılar olarak bizim de bu teknolojik gelişime ayak uydurmak adına yapmamız gereken şeyler listesinin başında, hazırladığımız çalışmaların sayfa çıktılarının daha esnek olmasını sağ-lamak geliyor.

Tasarımı web dokümanları haline getirirken, geleneksel grafik parçalama ve tablolama yöntemle-rinin kullanımı artık insanlar tarafından (eski /yeni değişiminin teknik taraflarının farkında olmasa da) kabul görmüyor.

Tablolar yerine kullanacağımız CSS uygulama teknikleri; biz tasarımcılarla çok büyük kolaylıklar getirmekle beraber, karmaşıklığı nedeniyle sektörde profesyonel ile amatör sınıfı daha da belir-ginleştirmiş durumdadır.

Tablolu tasarım dönemi sona erdiği için, artık CSS yardımıyla daha sık ve daha fazla kişiye erişe-bilen dokümanlar geliştiriliyor olacağız.

Bu kitapta, bir tasarımcının hazırladığı tasarımını CSS stillendirme yardımıyla hayatı geçirme ko-nusunda neredeyse her konuya değineceğiz.

Kitabın hazırlanışında, başta BilgeAdam Dijital Tasarım Eğitimleri Müdürü Başak Demir, Bilge-Adam Town Center Dijital Tasarım Eğitimleri Süpervizörü Gökhan Gökova ve manevi desteğini benden eksik etmeyen Gülsah Karasakal olmak üzere emeği geçen ve destek olan herkese son-suz teşekkürlerimi sunuyorum.

Onur Öztaşkıran

BilgeAdam

Web-Grafik Tasarım Eğitmeni, Eğitim Danışmanı

İçindekiler

Bölüm 1: CSS Nedir?	3
1.b - Niçin Tablosuz, CSS Destekli Tasarım?	4
1.c - CSS Versiyonları	6
Bölüm 2: CSS'in Web Dokümanlarına Atanması.....	9
2.a – CSS – Genel Bakış ve Uygulama Yöntemine (Syntax) Giriş	9
Seçiciler (Selektörler)	9
Selektör Özellikleri (Property) – Deklarasyon Blokları	9
Yorum Satırları	10
2.b – CSS'in Web Dokümanlarına Atanma Yöntemleri	10
2.b.i - Tarayıcı Standardı Stiller ve Kullanıcı Girişli Stiller	10
2.b.ii – Tasarımcı Yöntemi 1: Dahili Stillendirme	11
2.b.iii – Tasarımcı Yöntemi 2: Harici Stillendirme	12
2.b.iv – Tasarımcı Yöntemi 3: Import ile Stillendirme	12
2.b.v – Tasarımcı Yöntemi 4: Satır İçi Stillendirme	12
Hangisini kullanmalıyız?	13
2.c – CSS Seçicilere (Selektör) ve Özelliklere (Property) Giriş	13
2.c.i – Etiket (Tag) Selektörleri	13
2.c.ii – Class Selektörleri	16
2.c.iii – ID Selektörler ve Div'ler (Division)	17
2.c.iv Kalıtsallık, Child Selektörler ve Selektör Gruplama	19
2.c.v Selektörlerde Border (Çerçeve) Özelliği	21
2.c.vi - Selektörlerde Margin ve Padding	22
2.c.vii – Tarayıcılardaki Yorum Farkları	30
Bölüm 3: CSS Yardımıyla Web Tipografisi / Yazıtılı Biçimlendirme	35
3.a Web İçin Tipografi	35
Tipografi Neden bu kadar Önemli	35
Peki, Web Tasarımında Tipografinin Getirişi Ne Olacak?	35
3.b CSS Yardımıyla Şık Görünümlü Yazıtılı Oluşturmak	35
3.c – Tasarım ile Yazıtını Görsel Bütünlük Oluşturacak Şekilde Kullanmak	43
3.d – Daha Estetik Görünüm için “Görsel Yerdeğişim” Yöntemi	46
Alt Element İçerisini Gizleyerek “Görsel Yerdeğişim”	50

Bölüm 4: CSS ile Tasarımda Renk Uygulama.....	55
4.a Web Tasarımda Renk ve Kontrastın Önemi.....	55
4.b CSS içindeki Renk Tanımlamaları.....	57
1. Renk Tanımları.....	57
2. Hexadecimal (Sayısal) Renk Tanımları.....	59
3. RGB Değerleriyle Renk Uygulama.....	60
A Etiketi İçin Linklere Özel Class'lar.....	61
4.c Web Dokümanları İçindeki Elementleri CSS ile Renklendirmek.....	62
4.d CSS Renklendirme ile Derinlik Oluşturmak.....	65
4.e Kontrast Yönetimi	71
Bölüm 5: CSS ile Tasarımda Renk Uygulama.....	75
5.a Bir HTML Elementine Arkaplan Grafiği Atamak	75
<i>Bir HTML Elementine Arkaplan Grafiği Uygulama</i>	75
5.b Arkaplan Repeat (Tekrar) Özellikleri	78
5.c Arkaplan Pozisyonlandırma.....	81
5.d – Fixed (Sabit) veya Scroll (Kaydırılabilen) Arkaplan Özellikleri	84
Bölüm 6: CSS Konumlandırma 1: Float.....	89
6.a Kutuları (Block seviyesi elementler) Sola ve Sağa Yaslamak	90
<i>Float Eden Öğeleri Temizlemek / Kurtarıcı Kahramanlar (Clear Özelliği).....</i>	97
6.b Diğer Öğeleri Float Ettirmek	98
Bölüm 7: CSS ile Site Dolaşım Menüsü / Navigasyon Oluşturmak	107
7.a HTML'in Sıralı ve Sırasız Liste Öğeleriyle Menü Oluşturma Mantığı	107
7.b Liste Öğelerindeki Rakam ve İmleçleri Kaldırmak	107
7.c Liste İmleçleri Yerine Grafikler Kullanmak.....	116
7.d CSS ile Metin Tabanlı Yatay Navigasyon Oluşturmak	119
7.e CSS ile Yatay Sekmeli (Tabbed) Grafik Tabanlı Navigasyon Oluşturmak	124
<i>Sliding Doors / Kaydırılan Arkaplanlar Tekniğiyle Grafik Tabanlı Yatay Navigasyon</i>	127
Bölüm 8: CSS Konumlandırma 2: Relative, Absolute ve Fixed Konumlandırma	137
8.a Elementlerin Normal Akış ve Konum Düzeni	138
8.b Bağıl (Relative) Konumlandırma	139
8.c Mutlak (Absolute) Konumlandırma	143
8.d Hizalanmış İçerik Düzeni için Relative ve Absolute Konumlandırmayı Birlikte Kullanmak	146
8.e Sabit (Fixed) Konumlandırma.....	149

Bölüm 9: CSS ile Form Öğelerini Stillendirmek.....	155
9.a (x)HTML ile Erişilebilir Formlar Hazırlamak.....	155
9.b CSS ile Fieldset İçindeki Öğeleri Stillendirmek	157
9.c CSS ile Görsel Fieldset Izgarası (Grid) Oluşturmak-1	162
9.d CSS ile Görsel Fieldset Izgarası (Grid) Oluşturmak-2.....	164
9.e Form Butonlarını Stillendirmek.....	168
Bölüm 10: CSS ile Veri Tablolarını Stillendirmek	173
10.a (x)HTML ile Erişilebilir Tablo Yapısı Hazırlamak	173
10.b Tablo ve İçeriğini Stillendirmek.....	176
10.c Tablolarda Arkaplan Kullanmak.....	181
Bölüm 11: CSS ile Bir Tasarımı Hayata Geçirmek.....	187
11.a Tasarımın CSS Uygulamasını Planlamak: Tasarım Öğelerinin Element ve Stil Karşılıklarını Organize Etmek.....	188
11.b Dokümanı Stilsiz Olarak Hazırlamak.....	190
11.c Üst Kısmı Stillendirmek	196
11.d Orta Kısmı Stillendirmek.....	206
11.e İçerik Kısmını Stillendirmek	209
11.f Alt Kısmı Stillendirmek	218
EK A: Tam Liste CSS Özellik Tablosu	223



1

CSS Nedir?

1 CSS Nedir?

- Niçin Tablosuz, CSS Destekli Tasarım?
- CSS Versiyonları

CSS Nedir?

CSS / Cascading Style Sheets, ham HTML dokümanlarımızın barındırdığı içeriği ve genel doküman yapısını ziyaretçiye daha albenili ve estetik şekilde sunmak amacıyla kullanacağımız bir stillendirme dilidir.

CSS, HTML'den farklı bir dil uygulama yapısına (Syntax) sahip. Onunla HTML içinde olduğu gibi etiketler oluşturmak yerine etiketleri ve HTML dokümanları içindeki öğeleri seçip yapılarını değiştirebileceğimiz selektörler, seçiciler ve biçimleyiciler oluşturacağız.

CSS dokümanlarını HTML dokümanlarımız içine dahil ederek sayfalarımızın görünümünde köklü değişimler yapabilme şansı elde edeceğiz.

Biz tasarımcılar da dahil olmak üzere birçok insan her gün onlarca Web sitesini ziyaret ederiz. Peki, bir Web sitesinin görünen kısmı fizikal parçalar olarak neye tekabül ediyor, hiç düşündünüz mü?

Bu soruyu Internet'i gezen kullanıcılar soracak olursak, onlar bize Internet'in ve Web sitelerinin ihtiyaç duydukları bilgiyi görsel bir biçimde sunan dijital ortam olduğundan bahsedeceklerdir.

Soruya teknik açılarla cevap verecek olursak; Web sitelerini, ziyaretçilerin gözüne estetik bir biçimde sunulan içerik dokümanları olarak ele alabiliriz.

Biraz daha detaya inelim:

Bir Web sayfası dokümanına ait görsel anatomiyi incelemek istediğimizde, bu dokümanın teknik ve yapısal olarak **3** ana katmandan oluştuğunu görebiliriz. Bunlar yukarıdan aşağıya şekildeki gibidir:

1. Davranışsal Uygulamalar (Javascript, Flash vb.)
2. Görsel Sunuș (CSS/Stil dokümanı)
3. HTML (Doküman İskeleti ve İçeriği)

Bu üç katmayı doğrudan doküman HTML iskeleti içinde de barındırabiliriz. Keza tasarımcı ve geliştiriciler olarak CSS tabanlı tablosuz tasarım bir standart haline gelene kadar öyle yapıyorduk.

Ancak bu 3 katmayı ayrı parçalar halinde oluşturup düzenlemek kesinlikle ileri vadede sayfalarınızın güncellenmesi veya değiştirilmesi esnasında size hem hız, hem de esneklik kazandırıyor olacak.

Davranışsal uygulamalar, ziyaretçilerin Web sitesiyle olan etkileşim gücünü artırmak amacıyla eklenen katmandır.

Örneğin bir ögenin fare ile üzerine gelindiğinde açıklama gösteren kutular, bir sitede arama yaparken açılan otomatik kelime tamamlama yardımcıları, ileri ve geri butonlarına tıklandığında kayarak yenilenen resim galerileri ve bunlar gibi birçok etkileşimi ve kullanılabilirliği artıran yöntemleri davranışsal uygulamalar olarak tanımlayabiliriz.

Görsel sunuș, bu kitabın ana konusu olan CSS yardımı ile içeriği stillendirme katmanıdır. Ham içeriğinden ibaret olan HTML dokümanı, CSS dosyalarında tanımladığımız özelliklere sahip olacak şekilde estetik ve kullanışlı bir sunușla görüntülenirler.

İçerik katmanı ise dokümanımızın iskeletini oluşturan HTML'den ibarettir.

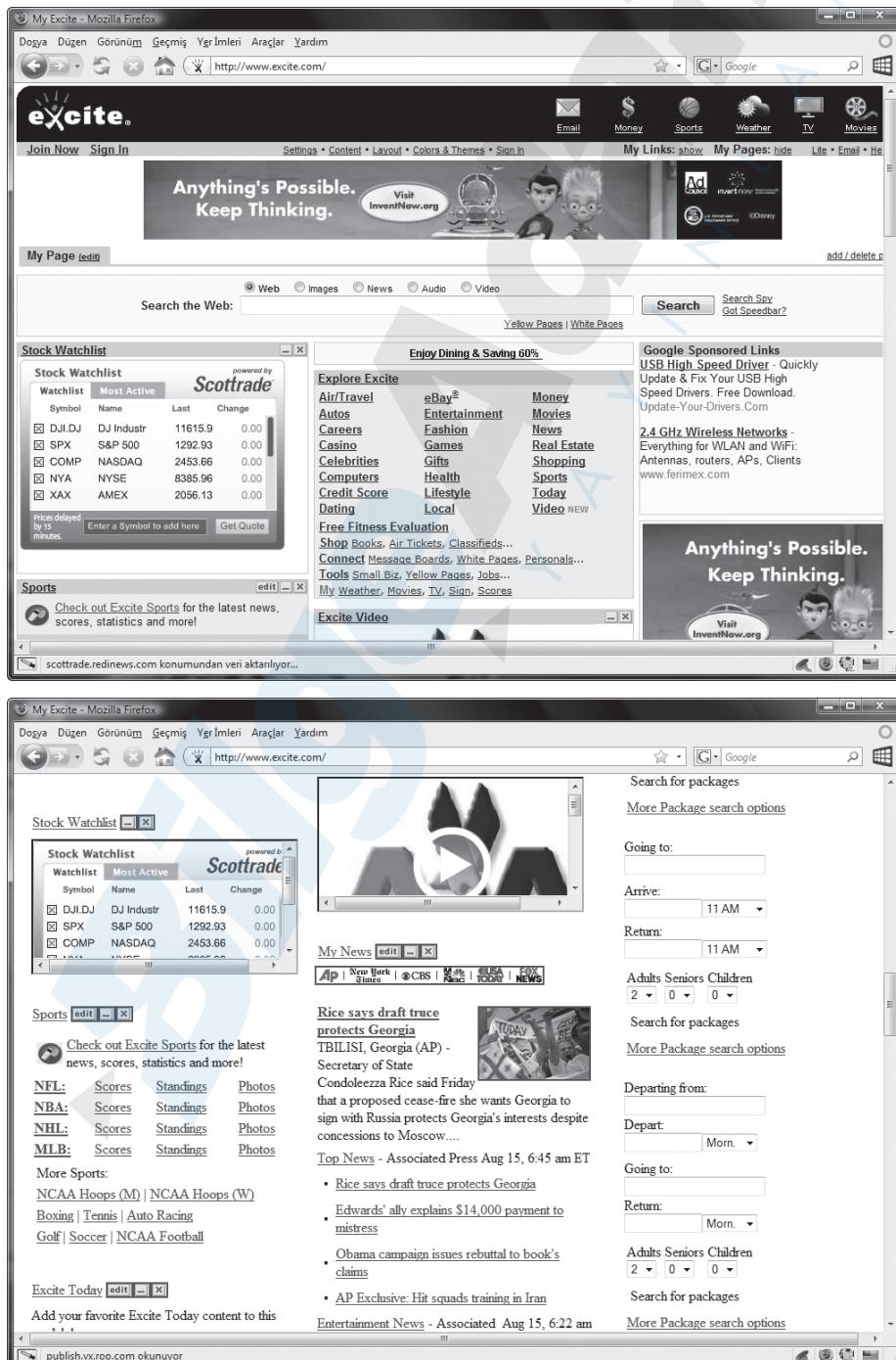
Burada bir noktaya dikkat çekmek istiyorum. Şu andan itibaren HTML (Hyper Text Markup Language) olarak söz edeceğimiz biçimleme dilinin belli yeni standart kurallara bağlı olduğunu düşünerek hareket edeceğiz. XML (extensible markup language) biçimleme dilinden gelen kurallar ile daha standart, evrensel ve tüm Internet tarayıcılarının sorunsuz tarayacağı bir dille, xHTML (Extensible Hypertext Markup Language) ile çalışacağız.

1.b - Niçin Tablosuz, CSS Destekli Tasarım?

Yakın vakte kadar tasarımcılar ve geliştiriciler olarak hazırladığımız Web tasarımlarını Internet üzerinde görüntülenecek dokümanlar olmaları için salt HTML'in kendi sahip olduğu obje biçimleme modelini ve davranışsal uygulamalar katmanını kullanıyordu.

Bir Web sitesinin görünümünü Photoshop veya Fireworks gibi görsel editörlerle hazırlayıp parçalara ayırdıktan sonra HTML kodu haline getiriyor, bu tasarımların HTML dokümanlarında görünüşü ve konumlandırılmasını ayarlamak için tablolar, yani `<table>` etiketlerini kullanıyordu.

Ancak ne var ki HTML içindeki `<table>` (Tabular Data) etiketinin asıl amacı konumlandırma yapmak ya da görsel elementleri bir arada tutmak değil, istatistik ya da liste yapısına sahip verileri belli bir düzende görüntülemektir. CSS'ten önce başka bir seçenekimiz olmadığı için yakın vakte kadar tablolar yardımıyla tasarımlarımızı Web sayfaları olarak hayata geçirdik.



Figür 1.b.1: Tablolu Excite.com Stilli ve Stilsiz Görünüm Farkı



Figür 1.b.2: Tablosuz CSS Tasarımı Cnet.com Stilli ve Stilsiz Görünüm Farkı

Bu dönem CSS'in Web dokümanları üzerinde etkin rol oynamaya başlamasıyla beraber kapanıldı.

CSS tabanlı biçimleme ile artık daha evrensel ve erişilebilir hale gelen sayfalarımız tasarım, içerik ve interaktivite (etkileşim) katmanlarından oluşuyor ve biz onları artık tablo içermeyen esnek yapıları sayesinde kolayca güncelleyebiliyor ve örneğin bugün mavi tonlardan oluşan bir Web sitesini birkaç saat içinde sarı tonlara sahipambaşka bir tasarım görünümüne kavuşturabiliyoruz.

CSS'in avantajları bununla sınırlı değil. HTML dosyalarına entegre edeceğimiz bu stil dokümanları ile;

- CSS stillendirme dosyası, gerekiğinde 1000 sayfalık bir Web sitesine atanarak ortak tasarım görünüme sahip olan,
- Internet Explorer, Firefox (www.getfirefox.com), Safari (<http://www.safari.com>) ve Opera (<http://www.opera.com>) gibi popüler Internet tarayıcılarında ortak standart ve hatasız görünüme sahip,
- Html dosyası ile bunun görünümünü içeren CSS stillendirme dosyaları ayrılığı için eski yöntemlere oranla %60 oranına kadar daha hızlı yüklenen ve daha az harcamaya neden olan bant genişliği sağlayan,
- CSS dokümanı ayrı bir dosya olarak saklandığı için tasarımsal değişiklik ve güncellemesi oldukça kolay ve hızlı bir şekilde gerçekleşen,
- Arama motorlarında daha iyi ve üst sonuçlar veren,
- Maliyetleri düşüren,
- Daha fazla insan tarafından ziyaret edilen

Web siteleri hazırlıyor olacağız.

1.c - CSS Versiyonları

İlk CSS versiyonu olan **CSS 1**, dünya genelinde kullanıcı tabanlı kodların standardizasyonundan sorumlu olan **Worldwide Web Konsorsiyumu (W3C – www.w3.org)** tarafından standart olarak onaylandı ve geliştiriciler tarafından kullanılmaya başlandı.

Bu ilk versiyon HTML dokümanları içindeki elementlere boşluk, yazıtılıp, hizalama, yaslama, renk değiştirme gibi temel biçimlemeler atamaya yönelik oldukça kullanışlı özellikler sunarak göze girse de, henüz tam anlamıyla bir Web dokümanının görünüm yapısını değiştirmeye olanak sağlamıyordu.

Bu nedenle bir süre boyunca tasarımcılar HTML dokümanlarındaki içeriğin bazı özelliklerini **CSS 1** yardımı ile (link renkleri değiştirme, doküman genelinde yazıtipi özellikleri belirleme vb.) biçimlerken, içeriğin yerleşim düzenini oluşturabilmek için tabloları (<table>) kullanmaya devam etti.

Tasarımcıların uzun bir süredir kullanılmayı dileydiği biçimleme özelliklerine sahip versiyon olan **CSS2** ise 1998 yılında ortaya çıktı.

Artık tabloların yerini alabilecek yerleşim düzeni kutuları hazırlanıp bunların özellikleri biçimleyebilmeye ve grafik editörlerinde hazırlanan tasarımları neredeyse birebir görünümde Web sayfaları halinde hayata geçirilemeye olanak sağlandı.

CSS2 versiyonu sadece bilgisayar ortamı için değil, aynı zamanda kiosk, cep telefonu ve avuç içi bilgisayar gibi platformlardan da Web sitelerinin görünümünü erişilebilir kılan ek özellikler sundu.

Ne var ki bazı biçimleme özelliklerinin uygulanışı sıkıntılı olduğu için **CSS 2** üzerinde birtakım düzenlemelere gidildi ve CSS daha kolay uygulama olanağı sunan "**CSS 2 seviye 1**" versiyonu olarak adlandırıldı. Biz yine de versiyon karmaşasını önlemek için CSS'in son halini versiyon 2 olarak ele alıyoruz.

Günümüzde ve bu kitabın içeriğinde CSS 2 versiyonu kullanılmaktadır.

Henüz tüm tarayıcılar tarafından desteklenmeyen ve kutu kenarlarını ovalleme, yazıların arkalaşına gölgeler verme, saydam objeler oluşturma gibi çok daha estetik yeni biçimleme olanaklarına da sahip olacak CSS 3 versiyonu ise yillardır geliştirilmeye devam ediyor. Bununla birlikte bu bahsettiğimiz özelliklerden bazılarını Firefox, Safari gibi modern tarayıcılar desteklemektedir.



2 CSS'in Web Dokümanlarına Atanması

2 CSS'in Web Dokümanlarına Atanması

- CSS – Genel Bakış ve Uygulama Yöntemine (Syntax) Giriş
- CSS'in Web Dokümanlarına Atanma Yöntemleri
- CSS Seçicilere (Selektör) ve Özelliklere (Property) Giriş

CSS'in Web Dokümanlarına Atanması

2.a – CSS – Genel Bakış ve Uygulama Yöntemine (Syntax) Giriş

CSS'in kod uygulama yöntemi, her uygulama dilinde olduğu gibi, kendine has kurallara sahiptir.

HTML dilinde dokümanda görüntülemek istediğimiz içeriğin türünü belirlemek için onlara has etiket oluşturma yöntemleri kullanırız. CSS dokümanlarında da HTML dokümanındaki içeriğin görünümü biçimlemek için CSS'e özel yazım yöntemini kullanıyor olacağız.

Seçiciler (Selektörler)

CSS dosyaları, içlerinde seçetör adı verilen seçenekler barındırır ve bu seçenekler sayesinde HTML dokümanı içindeki elemanların görsel ve yapısal özelliklerini üzerinde hakimiyet kurar.

Her seçetör adlandırılıp oluşturulduktan sonra süslü parantezle ({}) açılıp kapatılmaktadır.

Aşağıdaki örnekte HTML'deki paragraf (<p>) etiketini düzenlemek için açılmış bir etiket seçetörünü görüyoruz:

```
p {  
}
```

Selektör isimleri rakamla bitebilir ancak rakamla başlayamaz, Türkçe karakterleri ve özel karakterleri içeremezler.

Selektör Özellikleri (Property) – Deklarasyon Blokları

Bir HTML dokümanı içindeki elementlerin görünüm ve yapıları seçetörler tarafından içeriklerinde sahip oldukları özellikler (property) ile düzenlenlenebilir ve değiştirilebilirler.

Selektörlerin sahip oldukları özellik ve deklarasyon (tanımlama) blokları atandıkları HTML elementinin görünüm ve yapısı üzerinde değişiklikleri gerçekleştirir.

Her özellik kendine ait bir isme sahiptir ve bu özelliğe atanın değerler özelliğin yapısını belirler. Her özellik değerinden iki nokta üst üste (:) ile ayrılır ve her özellik satırı noktalı virgül (;) ile bitirilerek gerekliyorsa yeni özellik satırına geçilir.

CSS 2'de yaklaşık 9-10 kadar özellik grubu ve toplamda 120-130 adet özellik bulunmaktadır. Kitap içinde en sık kullandığımız birkaç tanesini kitabın sonundaki CSS Property indeksinde bulabilirsiniz.

Aşağıdaki örnekte özellikler arkaplanı sarı ve içerik metni yeşil olarak belirlenmiş bir <p> etiket seçetörünü görüyoruz:

```
p {  
background: yellow;  
color: green;  
}
```

Bazı özellikler birden fazla değere sahip olabilirler. Bu değerler birbirinden virgülle (,) ayrırlırlar.

Aşağıdaki örnekte **birden fazla yazılışı tercih değerleri tanımlanmış** bir font özelliğinin oluşturulduğu <p> etiket seçetörünü görüyoruz:

```
p {
  font-family: Arial, Helvetica, sans-serif, "Times New Roman";
}
```

Bu örnekte tarayıcı <p> etiketinin içinde barındırdığı metinleri (kişinin bilgisayarında halihazırda yüklüyse) öncelikle **Arial**, yüklü değilse **Helvetica**, bulunmuyorsa **sans-serif**, o da bulunmuyorsa son olarak **Times New Roman** yazıtipiyle görüntülemeye çalışacaktır.

Dikkat ederseniz Times New Roman değeri tırnak içindedir. Birden fazla kelimeye sahip değerler tırnak içinde yazılırlar.

Yorum Satırları

Bazen yüksek sayıda satırın bulunduğu, karmaşık ve onlarca selektörün bulunduğu bir CSS dokümanını uzun bir aradan sonra tekrar düzenlememiz gerekebilir.

Bu tip durumlarda hangi kuralı ne amaçla uyguladığımızı hatırlamak veya dosayı düzenleyecek bir başkasının CSS dokümanı içinde kaybolmasını önlemek amacıyla kuralların veya yazılanların ne amaca hizmet ettiğini tanımlayıcı, işlevleri hatırlatmaktan başka hiçbir fonksiyonu olmayan yorum satırlarını kullanabiliriz.

Bir yorum satırı /* ile başlar ve */ ile biter:

Örnek:

```
/* HTML dokumani icindeki paragraflari oncelikle Arial yazitipiyle
goruntule */

p {
  font-family: Arial, Helvetica, sans-serif, "Times New Roman";
}
```

2.b – CSS'in Web Dokümanlarına Atanma Yöntemleri

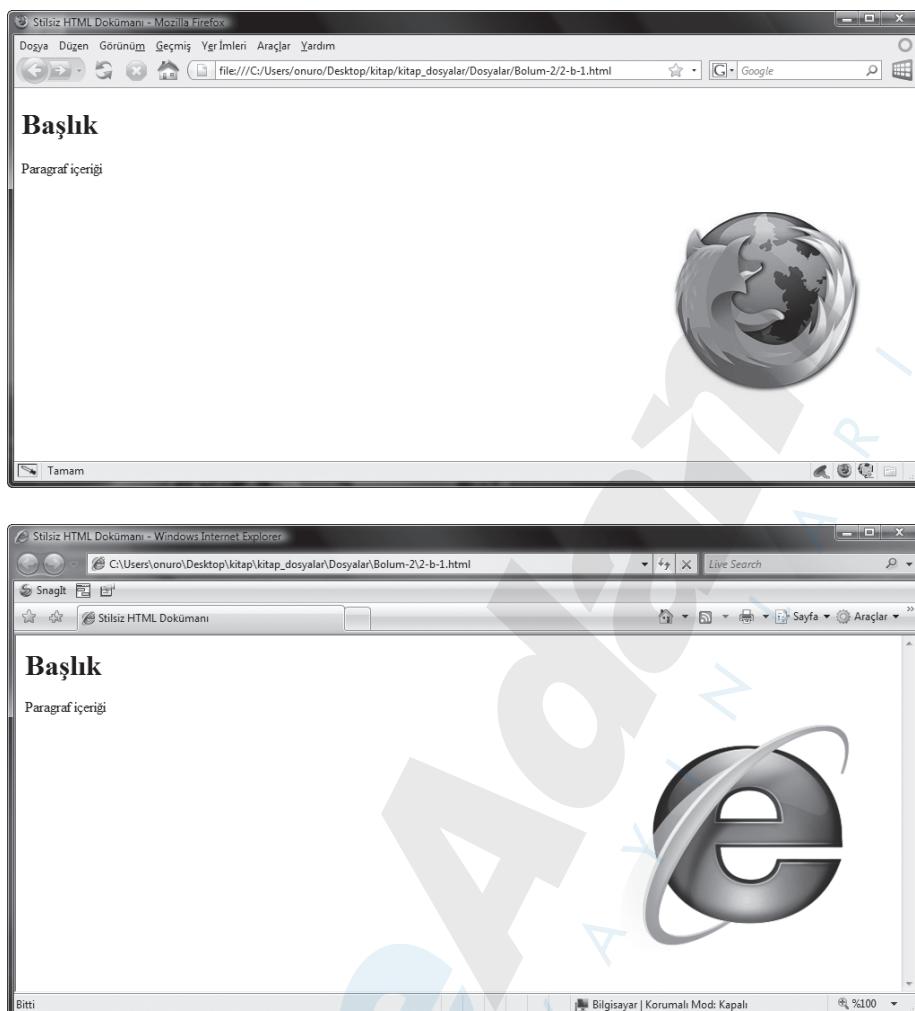
2.b.i - Tarayıcı Standardı Stiller ve Kullanıcı Girişli Stiller

Tarayıcı Standardı Stiller

Internet tarayıcılar programlanırken bize Web sitelerine ziyaretlerimiz esnasında en yüksek verim ve görünüm kalitesini ulaştıracak şekilde hazırlanırlar. Böylece bazen hazırlayıcısı tarafından özen gösterilmemiş bir Web sitesinin içeriğini de fazla sıkıntı çekmeden gezebilmemizi sağlarlar.

Her Internet tarayıcısının içinde hiç stil verilmemiş bir HTML dokümanın nasıl görüntüleneceğine dair kuralları içeren bir stil dokümanı bulunur ve stilsiz bir sayfayı ziyaret ettiğimizde bize o sayfayı kendi varsayılan stil kurallıyla gösterir (bu stil kuralları neredeyse bütün tarayıcılarda aynıdır)

Figür 2'de Firefox ve Internet Explorer tarayıcılarında stilsiz ham bir HTML dokümanın standart olarak nasıl görüntüldüğünü görüyoruz:



Kullanıcı Girişli Stillendirme

Yeni nesil birçok tarayıcı ziyaretçilere gezdikleri Web sitelerinin görünümünü CSS yardımıyla biçimlemelerine olanak kılan eklentiler sunmaktadır.

Örneğin Firefox tarayıcısına eklenti olarak kurulabilecek Greasemonkey veya Firebug eklentileri ile google.com.tr ana sayfasının görünümünü CSS ile biçimleyerek değiştirebiliriz.

Kullanıcı girişli stillendirme yalnızca ziyaretçinin kendi bilgisayarına ait biçimlemeleri kapsar, site-lerin diğer ziyaretçilerinin görüntüleyeceği biçim etkilemez.

2.b.ii – Tasarımcı Yöntemi 1: Dahili Stillendirme

CSS kodlarını hazırladığımız Web dokümanlarına dahil etmenin 3 yolu vardır. Web dokümanlarında bu 3 yol da kullanılabilsse de, her yöntemin kendi temel amacı bulunmaktadır.

Dahili Stillendirme: Bir HTML dokümanı içerisinde `<head>` etiketinin açılışı ile kapanışı `</head>` arasına açacağımız `<style>` etiketi ile o dokümana CSS stillendirmesi ekleyebiliriz:

```
<DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns= "http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html charset=utf-8"/>
```

```

<style type="text/css">
/* CSS KODLARI BURADA */
</style>

</head>
<body>...

```

2.b.iii – Tasarımcı Yöntemi 2: Harici Stillendirme

Harici Stillendirme: Bu yöntemde ayrı bir stil dokümanı hazırlanıp “.css” uzantısıyla kaydedildikten sonra bu dosya HTML dokümanına <head> etiketinin açılı ile kapanışı </head> arasına açılacak <link /> etiketi içinde yapılan css doküman yolu tanımlaması ile dahil edilir:

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns= "http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html charset=utf-8"/>

<link rel="stylesheet" type="text/css" href="stil.css" />

</head>
<body>...

```

2.b.iv – Tasarımcı Yöntemi 3: Import ile Stillendirme

Import Yöntemi ile Stillendirme: Bu yöntem harici stillendirme ile aynı maksadı taşımakla beraber uygulama yöntemi harici stillendirmeden ufak bir farkla ayrılır. Bu teknikte dahili css dosyası oluşturur gibi <style> etiketi açılır ancak içeriğine selektörler ve özellikleri girmek yerine import özelliğle harici css dosyasının yolu girilir:

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns= "http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html charset=utf-8"/>

<style type="text/css">
@import url(stil.css);
</style>

</head>
<body>...

```

2.b.v – Tasarımcı Yöntemi 4: Satır İçi Stillendirme

Satır İçi Stillendirme: Bu yöntemde ayrı bir stil dokümanı hazırlanıp “.css” uzantısıyla kaydedildikten sonra bu dosya HTML dokümanına <head> etiketinin açılı ile kapanışı </head> arasına açılacak <link /> etiketi içinde yapılan css doküman yolu tanımlaması ile dahil edilir:

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns= "http://www.w3.org/1999/xhtml">

```

```

<head>
<meta http-equiv="Content-Type" content="text/html charset=utf-8"/>

<link rel="stylesheet" type="text/css" href="stil.css" />

</head>
<body>...

```

Hangisini Kullanmalıyız?

Tek bir HTML dokümanının görünüm özelliklerini biçimlemek için dahili stillendirmemiz, birden fazla HTML dokümanının ortak görünüme sahip olması için tek merkezli harici stillendirme dokümanı kullanmamız, bazen de bu ikisinin yetersiz kaldığı anlık durumlarda satır içi stillendirmeyi yapmamız gerekebilir.

Tahmin edeceksinizdir ki, anlık ve mecburi durumlar haricinde satır içi stillendirmeyle dolu bir HTML dokümanı hazırlamanın aslında eski karmaşık ve düzenlenmesi sıkıntılı yaratacak tasarım yapısından pek farkı kalmayacaktır.

Yine de sonuç itibariyle atama yöntemleri tamamen sizin kişisel inisiyatifinizde. İleriki örneklerimizde bu hususla ilgili uygulamaları görüyor olacağız.

2.c – CSS Seçicilere (Selektör) ve Özelliklere (Property) Giriş

CSS kod uygulama yöntemleriyle ilgili bölümde belirtmiş olduğum gibi; CSS selektörler bizim HTML dokümanımızdaki elementleri seçerek onların dilediğimiz özelliklerle görüntülenmesini veya doküman içinde dilediğimiz pozisyonlara dilediğimiz boyutlarla yerleşmelerini sağlarlar.

Her zaman öncelikle Web sitemizin veya HTML sayfamızın içeriğini HTML kodları yardımıyla, herhangi bir stillendirme yapmadan oluşturacağız. Bu şekilde hiyerarşimizin içerik ve iskelet katmanınız hazırlanır.

Daha sonra CSS kodlamasına geçilerek HTML dokümanının görünümünü etkileyebilecek olayları belirlemek için selektörler oluşturulur ve yavaş yavaş sayfamızın görünümü ve görsel yapısı üzerinde fark oluşturmaya başlarız.

Selektörler 3 ana türde olmakla beraber bazı özel tanımlamalar için gruplanmış ve alt element seçici gibi ara selektör tipleri de vardır.

Şimdi bu 3 ana selektörü ve uygulama amaçlarını görelim. İlk olarak HTML dokümanındaki etiketleri seçerek tüm etiketlere ortak özellik atamamızı sağlayacak **etiket (tag)** seçimleri ve genel selektör yapılarını tanıyalım:

2.c.i – Etiket (Tag) Selektörleri

Etiket selektörleri, HTML dokümanının içindeki bir etiketi seçerek doküman içinde o etiketin geçtiği her noktada etiketin görünüşü üzerinde değişiklik yapma olanağı sağlarlar.

Şimdi yeni bir HTML dokümanı oluşturalım ve CSS'in etiket selektörleri ile bu dokümanın görünümünü yavaş yavaş biçimlemeye başlayalım:

```

<DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns= "http://www.w3.org/1999/xhtml">
<head>

```

```

<meta http-equiv="Content-Type" content="text/html charset=utf-8"/>
<title> Merhaba Dünya!</title>

<style type="text/css">
/* az sonra buraya css kodlarını gireceğiz*/
</style>

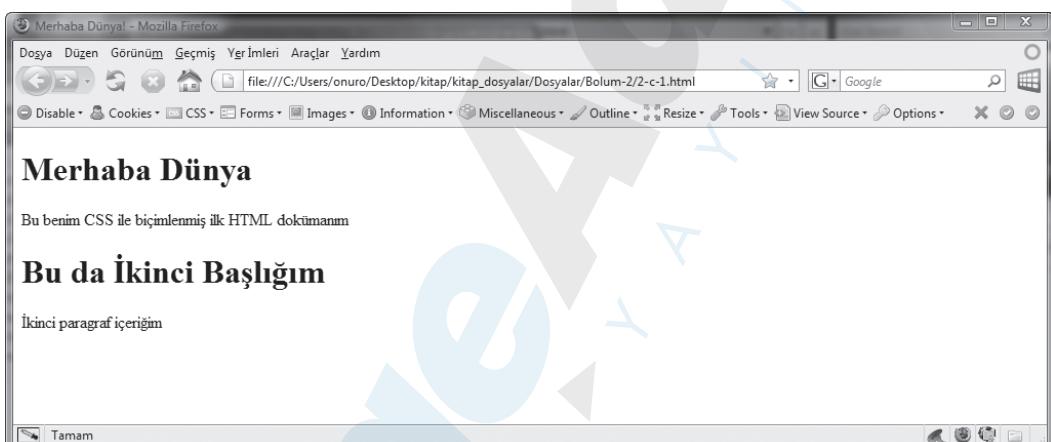
</head>
<body>

<h1> Merhaba Dünya </h1>
<p> Bu benim CSS ile biçimlenmiş ilk HTML dokümanım </p>

<h1> Bu da İkinci Başlığım </h1>
<p> İkinci paragraf içeriğim</p>

</body>
</html>

```



Figür 2.c.1

Pek albenili görünmüyör değil mi? Bunu CSS yardımıyla değiştireceğiz.

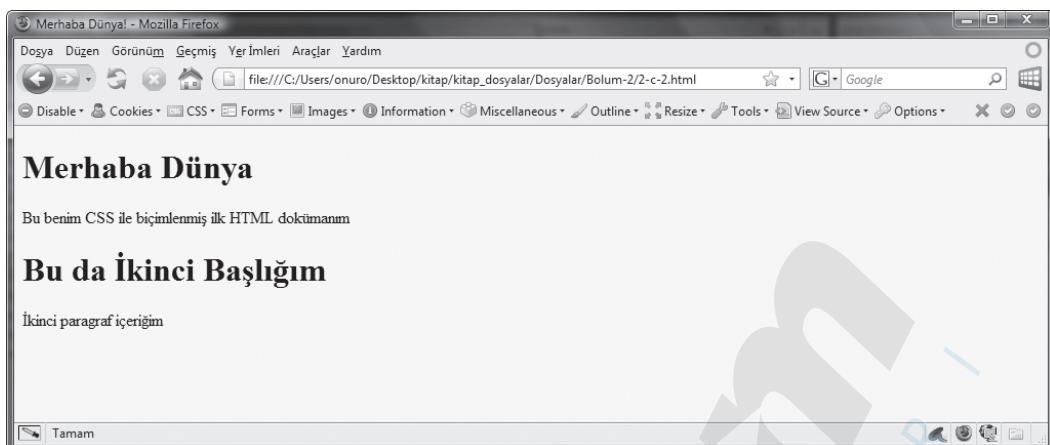
Bir etiket selektörü etiketin adının selektör olarak açılmasıyla oluşturulmuş olur. Herhangi bir imlece ihtiyaç duymazlar.

Aşağıdaki örnek bize HTML'deki <body> etiketinin biçimlendirerek sayfanın tamamının sarı arkaplana sahip olmasını sağlayacak:

```

...
<head>
<style type="text/css">
body {
    background: yellow;
}
</style>
<head>
...

```



Figür 2.c.2

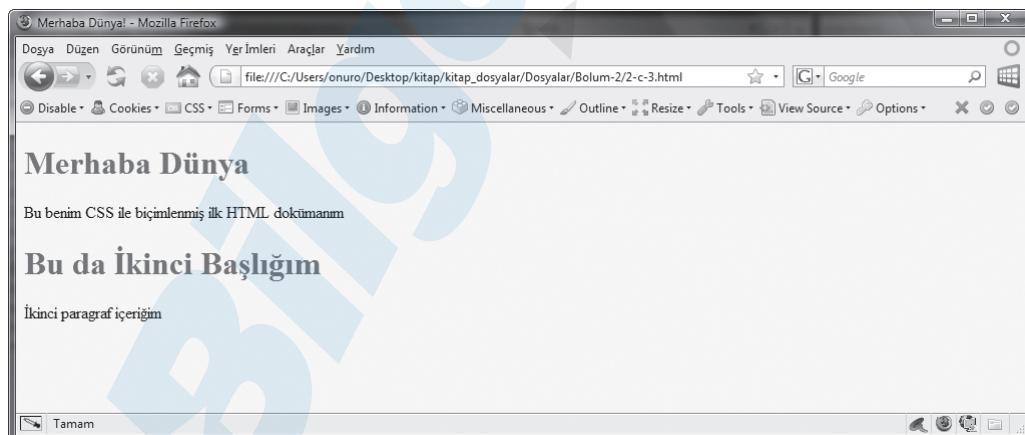
Bir noktayı tekrar edelim. Etiket selektörleri, kullanıldıkları andan itibaren HTML dokümanında adı geçen etiketin yer aldığı her noktada hakimiyete sahip olacaktır.

Bu da şu demek:

Örneğin HTML içindeki h1 elementini metin rengi yeşil olacak şekilde etiket selektörü ile biçimleyeceğiz olursak, doküman içinde yer alan tüm h1 başlıklar stillenmiş olacaktır.

```
body {
background: yellow;
}

h1 {
color: green;
}
```



Figür 2.c.3: Yapılan değişikliğin sonucu

Gördüğü gibi doküman içindeki tüm h1 boyutundaki başlıklar yeşil yazıtipi özelliğine sahip oldu.

Sonuca gelecek olursak; etiket selektörleri çoğunlukla HTML dokümanımızın içeriğini biçimlemek maksatlı kullanırız.

2.c.ii – Class Selektörler

Etiket selektörlerini işlerken onların doküman içinde kullanıldıkları her noktada etki ettiğinden bahsetmiştim.

Peki, örneğin ya biz h1 başlıkların tamamını yeşil renk gösterirken bunlardan 1 tanesini hariç tutarak mor metin rengiyle göstermek istiyorsak?

Etiket selektörleriyle stillendirme gerçekleşse de yukarıda bahsettiğim konu için yeterli olmayacak gibi görünüyor. İşte burada, etiket selektörlerin yetersiz kaldığı noktada **class selektörleri** kullanıyoruz.

Class selektörleri etiket selektörlerden farklı olarak, kendi belirlediğimiz isimlendirmelerle, başına nokta (.) tanımlayıcısı ile açabiliyoruz.

Yine etiket selektörlerinden farklı olarak, CSS içerisinde class selektörü açmış olmak, biçimlemeyi gerçekleştirmez. Oluşturduğumuz class selektörünün adını, HTML dokümanımızda atamak **istediğimiz elemente HTML olarak uygulamak** gereklidir.

Şimdi, dokümanımızdaki tüm h1 başlık etiketlerini yeşil metin rengiyle tutarken bunlardan bir tanesini mor renkle görüntülemek için “**morbaslik**” adını vereceğimiz bir class selektör ile CSS kodlarımıza ekleme yapalım:

```
...
h1 {
  color: green;
}
.morbaslik {
  color: purple;
}
...
```

Şimdi, oluşturulan “**morbaslik**” adlı class selektörünü HTML dokümanımızdaki h1 elemanlarından bir tanesine uygulamak istersek:

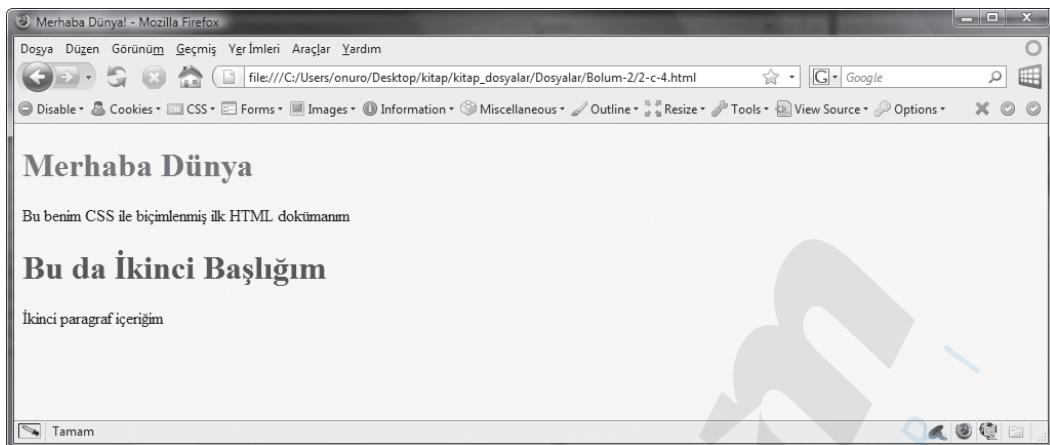
```
...
<body>

  <h1> Merhaba Dünya </h1>
  <p> Bu benim CSS ile biçimlenmiş ilk HTML dokümanım </p>

  <h1 class="morbaslik"> Bu da ikinci Başlığım </h1>
  <p> İkinci paragraf içeriğim</p>

</body>
</html>
```

Bu değişikliğin sonucu şekildeki gibi olacaktır.



Figür 2.c.4

Böylece yavaş yavaş HTML dokümanımızın içeriğini etiket selektörler ile biçimlerken, onların yetersiz kaldığı bir veya daha fazla noktada class selektörleri kullanarak biçimleme yapabiliyoruz.

Bir iki unsuruunu çizelim:

- Class selektörler bir HTML dokümanındaki elementlere birçok kez atanabilir, doküman içerisinde tek tek kullanılabılır.
- Etiket selektörlerin yetersiz kaldığı noktalarda class selektör kullanıyor olacağım demiştim. Ancak, bunun dozunu kaçırır; etiket selektörlerin yetmediği her noktada class selektör açmaya başlarsak dokümanımızı gereksiz yere şişirmeye başlarız.

Bu muhtemel yanlış algıyı ortadan kaldırmak için ileriki konularda alt element selektörleri göreceğiz.

- Etiket ve class selektörler ağırlıklı olarak HTML dokümanının içeriğini ve sık kullanılacak içeriğin görünüm özelliklerini belirlemek için kullanılır. Yapısal ve yerlesimle alakalı özellikleri belirlemek için az sonra geleceğimiz ID selektörleri kullanıyor olacağınız.

2.c.iii – ID Selektörler ve Div'ler (Division)

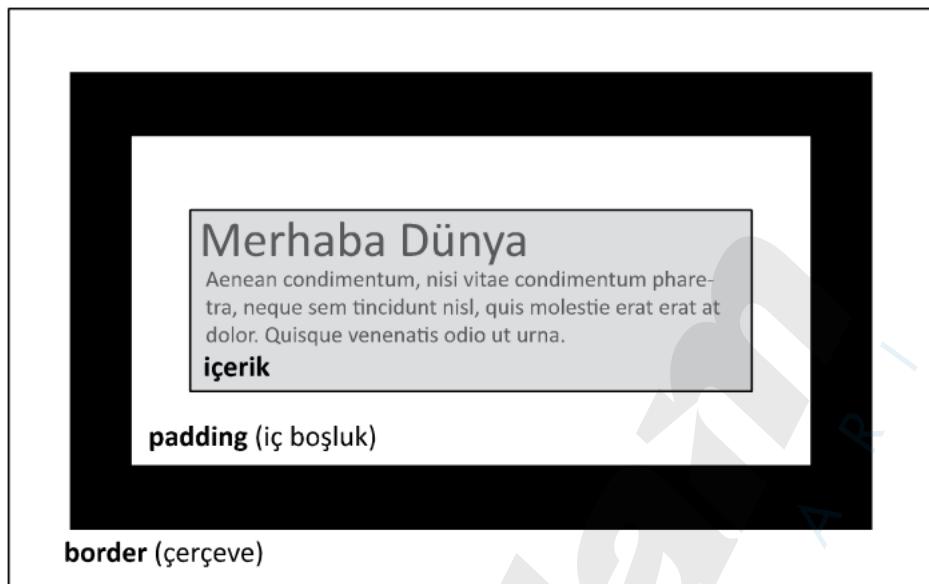
HTML dokümanımızın sahip olduğu içeriği biçimlemek için etiket ve class selektörleri kullanırız demiştim. Peki, Photoshop'ta veya benzeri bir programda oluşturduğumuz görsel tasarım yapısını CSS'le nasıl hayatı geçireceğiz?

Burada, ID selektörler devreye giriyor. HTML'in kutu modeli (box model) konusunda bilgi sahibi olanlar bahsedeceklerime nispeten daha aşina olacaklar; zira CSS HTML'e bağlı kutu modelindeki yapıyı biçimlendirecek selektör özelliklerine sahiptir.

CSS Box Model - Kutu Modeli

HTML içinde oluşturduğumuz paragraflar, h1'den h6'ya kadar olan başlıklar, div elementi ve tablolar, kısaca tüm block (kutu) seviyesi elementler kutu modeline dahildir.

Bu, bizim CSS yardımıyla block seviyesi elementlerin sahip olduğu standart iç boşluk (padding), dış boşluk (margin), çerçeve (border), Arkaplan (background) gibi özelliklerini kolayca değiştirebilmemize olanak sağlar.

*CSS Kutu Modeli*

ID Selektör Yapısı

Tekrar edelim; HTML içeriğimizin görünümünü düzenlemek için etiket ve class selektörleri, yapisal yani kutu yerleşim ve sayfa düzenini oluşturma maksatlı biçimlemeler içinse ID selektörleri kullanıyor olacağız.

ID selektörler oluşturma yöntemleri itibariyle class selektörler ile birebir aynıdır. Tek farkla:

ID selektör tanımlarken bu kez belirteç olarak diyez (#) işaretini kullanırız.

Aşağıda, “**anakutu**” adı verilerek açılmış bir ID selektör görüyoruz:

```
...
#anakutu {
    width: 750px;
    border: 5px solid black;
    margin: 0 auto;
    background:white;
}
```

Buradaki tanımlamalara göre; “**anakutu**” ID’sinin atanacağı element 750 piksel genişliğe, 5 piksel siyah tek ton çerçeveye, beyaz arkaplana sahip olacak ve bir üst elementine oranla yatay olarak tam ortaya yerleşecek şekilde hizalanacaktır.

Şimdi, “**anakutu**” ID selektörünü HTML dokümanı içindeki bir elemente atayalım:

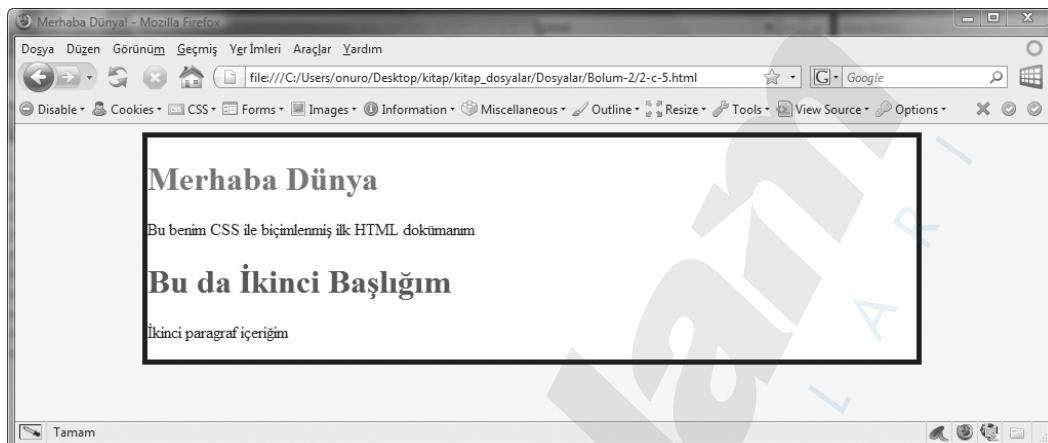
```
...
<body>

<div id="anakutu">
<h1> Merhaba Dünya </h1>
<p> Bu benim CSS ile biçimlenmiş ilk HTML dokümanım </p>

<h1 class="morbaskanlik"> Bu da ikinci Başlığım </h1>
<p> ikinci paragraf içeriğim</p>
```

```
</div>
</body>
</html>
```

Bu değişikliğin sonucu şekildeki gibi olacaktır.



Figür 2.c.5

Göründüğü gibi, dokümanımız artık sahip olduğu standart ve yalın HTML görüntüsünden uzaklaşıp nispeten daha farklı bir hal almaya başladı. CSS içindeki özellikleri kullanarak binlerce görsel varyasyon üretmek mümkün.

ID selektörler HTML dokümanı içinde yalnızca bir kez uygulanabilirler.

Div'ler – Division'lar

Fark ettiğiniz üzere, bir önceki örneğimizde “anakutu” id selektörünü HTML içinde bir `<div>` etiketi üzerine atadık.

HTML içindeki bu elementin amaçlarına biraz göz atalım:

Div (division) elementleri HTML dokümanınız içindeki block level (kutu seviyesi) elementlerdir. Div'leri dokümanınızdaki yapısal bölümleri tanımlamak için kullanırız. Div'ler bizim sitemizin içeriğini oluşturacak tüm öğeleri kapsayabilir (aynı diğer tüm block level elementlerin yaptığı gibi).

Ancak diğer block level elementlerden farklı olarak, div'ler kendilerine CSS tarafından bir stil atanmadan hiçbir görsel değişiklik getirmezler.

Yani bir div açıp içine içeriği girerseniz sonuçta elde edeceğiniz görüntü o içeriğin div içinde olmadan alacağı görüntüyle aynıdır, herhangi bir görsel değişim olmayacağındır.

Dolayısıyla div'leri, yani bu hayali kutuları daha çok CSS ile oluşturmak istediğimiz hiyerarşik fiziksel yapılar için kullanacağız.

Bir önceki örnekte bahsettiğimiz hayali kutu etkisi için “container” adlı bir id selektör oluşturup dokümandaki bir div'e atamıştık. Div'ler çoğunlukla id selektörler ile birlikte anılsalar da, gerekli her durumda class selektörleri de div'ler üzerinde kullanabiliriz.

2.c.iv Kalıtsallık, Child Selektörler ve Selektör Gruplama

Kalıtsallık

Gözleriniz ne renk? Kahverengi, yeşil, mavi? Boyunuz anne veya babanızinki gibi oldukça uzun veya benzer oranlarda mı? Esmer misiniz, sarışın mı? Bu tip soruları artırmak mümkündür.

Hepimizin bildiği gibi, bütün bunlar ebeveynlerimizden gelen kalıtsal özellikler. Pekala, konuya alakamıza bağlayayım:

HTML dokümanındaki elementler daima birbiriyle bağlantılıdır. CSS ile biçimleyeceğimiz dokümandaki birincil özelliğe sahip ana elementin sahip olduğu bazı özellikler, altındaki diğer elementin de özelliklerini etkiler, değiştirir.

Bu şu demek; eğer biz `<body>` etiketine ait bir etiket selektörü açıp yaztipini “**Verdana**” olarak belirlersek body elementinin altındaki tüm elementlerin (örn: paragraf, tablo, başlık, link vb.) de yaztipi “Verdana” olacak, yani ebeveyni olan body’den bu özelliği alacaklardır.

Bu bize ebeveyn elementlerden birine biçimleme atadığımızda, altındaki elementleri de biçimlemeye gerek kalmadan düzenleme yapabilmemizi sağlar. Dolayısıyla yazitipi örneğinde, body’de yaptığımız “Verdana” değişikliğini diğer elementlere de atamak zorunda kalmayız.

Ta ki, biz alt elementlerden birinin “Verdana” yaztipine sahip **olmamasını** isteyene kadar. Tahmin edeceğiniz üzere, bu durumda yapmamız gereken şey yazitipi özelliği tüm dokümandan farklı olmasını istediğimiz element için yeni bir selektör açarak gerekli değişikliği yapmak olacak.

Tüm CSS özellikleri (property) kalıtsallığa sahip değildir. Yazitipi (`font`) bunlardan biriydi; bunun gibi kalıtsallığa sahip özellikleri uygulamalarınız esnasında deneyerek görebilirsiniz

Child Selektörler (Alt Element Seçicileri)

Hatırlarsanız, class selektörleri incelerken bir şeyin altını çizmiştim:

İçeriğimizi düzenlerken her sıkıştığımız noktada class selektör kullanıyoruz.

Örneğin dokümanımızdaki tüm linklerin rengini kırmızı olarak belirledik ancak paragraf elementleri içinde yer alan linklerin doküman genelinden farklı olarak yeşil renkte görüntülenmesini istiyoruz. Normal şartlarda, yeşil renkli linklere özel bir class selektör oluşturdukten sonra değişiklik gereken her link için atamalarını yaparak bu sıkıntımızı giderebiliriz.

Ancak onun yerine, daha efektif bir yöntem olan child selektörleri de kullanabiliriz.

Child selektörler, bir selektörün atandığı elementin hiyerarşik olarak altında veya içinde bulunan elementlerin, yani alt elementlerin biçimlenmesi için kullanılabilirler.

Örneğin; paragraf içindeki linkler, tablolar içindeki resimler, #kutu id selektörünün atandığı HTML elementi içinde bulunan paragraflar gibi öğeler onlara özel selektör açılmasına gerek kalmadan child selektörlerle biçimlenebilirler.

Child selektörler seçilecek alt elementin başına boşluk ” “ veya ”>” belirteci konarak onun ebeveyni olan selektör gelecek şekilde oluşturulurlar.

Aşağıdaki örnekte, yalnızca paragraf (`<p>`) etiketi içindeki linklerin özelliği kırmızı renkte görünecek şekilde değişecek, bunun dışındaki linklerde herhangi bir görsel değişiklik gözlenmeyecektir.

```
p>a {  
    color:red;  
}
```

Başa bir örneğe daha göz atalım. Aşağıdaki uygulama `#anakutu` id selektörünün atandığı elementin içinde bulunan paragraflarda bulunan linklerin kırmızı metin rengiyle görüntülenmesini sağlayacaktır. Bu değişim, anakutu içindeki paragraflar haricinde bulunan linklerin rengini veya başka bir özelliğini etkilemeyecek, yalnızca anakutu içindeki paragrafların kapsadığı linkler için geçerli olacaktır.

Uygulama esnasında “>” yerine bu kez boşluk “ ” kullanarak selektörü açtıgima dikkatinizi çekmek istiyorum (bu, daha sık kullanılan bir yöntemdir).

```
#anakutu p a {
    color:red;
}
```

Selektör Gruplama

Bazen, HTML dokümanımızdaki bazı elementlerin hepsinin birtakım ortak özelliklere sahip olmasını isteyebiliriz. Örneğin, h1'den h6'ya kadar olan tüm başlık etiketlerinin altının çizili olması gibi.

Bu tip durumlarda selektör gruplamak bize yardımcı olacaktır. Ortak özelliklere sahip olacak selektörler, birbirlerinden virgülle “,” ayrılarak toplu bir selektör grubu olarak açılır ve oluşturulurlar.

Aşağıdaki örnekte tüm başlık etiketlerinin altı çizili şekilde görüntülenmesini sağlayacak selektör grubunu görüyoruz.

```
h1, h2,h3,h4,h5,h6 {
    text-decoration: underline;
}
```

Eğer selektör gruplamayı kullanmamış olsaydık, her başlık etiketi için selektörlerini açarak altı çizili (underline) özellik değerini tekrar tekrar vererek gereksiz yere CSS dokümanımızı şişirmiş olacaktık.

2.c.v Selektörlerde Border (Çerçeve) Özelliği

Kutu seviyesi (**block level**) elementlerin biçimlenmesi esnasında, bazen şıklığı artırmak, bazen de ögenin sahip olduğu hatları daha iyi görebilmek için obje çerçeve (border) özelliğini kullanabiliriz.

border özelliği yardımıyla bir HTML elementinin sahip olacağı çerçeve rengini, kalınlığını ve stilini belirleyebiliriz:

```
p {
    border: 1px solid red;
}
```

Bu uygulama, dokümandaki tüm paragrafların 1 piksellik tek ton kırmızı dış hat çizgisine sahip olacağını belirtir. Kullandığımız yöntem border özellik grubuyla biçimlemeydi ve burada sıralama öncelikle çerçeve **kalınlığı**, sonra **stili**, sonra da **renki** şeklidindedir.

Bu üç özellik kendi adlarıyla ayrıca da kullanılabilir. İlgili özellikler aşağıdaki gibidir:

- **border-width:** çerçeveyenin kalınlığı
- **border-style:** çerçeveyenin stili
- **border-color:** çerçeveyenin rengi

border-width özelliğinin alabileceği uzunluk değeri piksel (px) veya aşağıdaki standart değerleri alabilir:

- **thin:** ince
- **medium:** orta kalınlık
- **thick:** kalın

`border-style` ise aşağıdaki değerleri alabilir:

- **dashed**: tireli
- **dotted**: noktalı
- **double**: çift hatlı
- **groove**: genişleyen
- **hidden**: gizli
- **inset**: içe göçük
- **none**: çerçeveye görüntülenmez
- **outset**: dışa çıkış
- **ridge**: kırıklı
- **solid**: tek ton/renk

`border-color` ise aşağıdaki değer türlerini alabilir:

- renk adı
- hexadecimal renk kodu
- rgb renk kodu
- transparan

Çerçevenin alacağı kalınlık oranını, rengini ve stilini 4 yönde ayrı ayrı değiştirmek de mümkündür.

```
p {
    border-top: 1px dashed black /* 1 piksel siyah tireli üst
                                   çerçeve*/;
    border-right: 1px dotted red /* 1 piksel kırmızı noktalı sağ
                                   çerçeve*/;
    border-bottom: 1px solid black /* 1 piksel siyah tireli alt
                                   çerçeve*/;
    border-left: 1px dashed red /* 1 piksel kırmızı tireli sol
                                   çerçeve*/;
}
```

2.c.vi - Selektörlerde Margin ve Padding

Tarayıcı standardı stillendirmede, ziyaretçi dokümanı gezerken dokümandaki elementlerin bazı varsayılan özelliklerle görüntülendiğinden bahsetmiştim.

Bu varsayılan değerlerden kutu modelindeki özellikle dış boşluk (margin) ve iç boşluk (padding) çoğunlukla bize CSS yardımıyla tasarımlarımızı hayatı geçirirken sıkıntılardan yaratabilecek, tarayıcı farkı görüntülemeye neden olabilecek bazı handikaplara sahip. Hem margin ve padding özelliklerine giriş yapmış olalım, hem de bahsettiğim muhtemel handikapları nasıl yok edebileceğinize yönelik birkaç durum ve yöntemi anlatmak istiyorum.

2.vi.a Dış Boşluk – Margin

Margin, CSS ile seçilmiş olan bir HTML elementinin diğer elementlere olan uzaklığını ve 4 yönde mesafesini (üst, alt, sol, sağ) ayarlamamızı sağlayan bir özelliktir. Özellik yapısı aşağıdaki gibidir:

- **margin-top:** üst boşluk mesafesi;
- **margin-left:** sol boşluk mesafesi;
- **margin-right:** sağ boşluk mesafesi;
- **margin-bottom:** alt boşluk mesafesi;

Bu 4 yöne girebileceğimiz uzunluk değeri türleri ise aşağıdaki şekildedir:

- Px (piksel) cinsinden uzunluk; **örn:**

```
p {
  margin-left: 15px;
}
```

- Percent (yüzde) cinsinden uzunluk; **örn:**

```
p {
  margin-top: 50%;
}
```

- Auto (otomatik değer) – genelde uygulanan ögeyi üstündeki elementin içinde dikey olarak ortalamak için kullanılır (bu durum için elbette o elementin sabit ve sınırlı bir genişliğe sahip olması gereklidir. %100 genişliğe sahip bir ögeyi ortalamak tabii ki mümkün olmayacaktır); **örn:**

```
#anakutu {
  width: 800px;
  margin-left: auto;
  margin-right: auto;
}
```

4 yöne ayrı ayrı özellik açarak biçimleme yapmak yerine, margin özellik grubunu kullanarak uygulama yapmak da mümkündür:

```
#anakutu {
  margin: 10px 10px 10px 10px;
}
```

Pekala, şimdi bu ne demek? Bu şu demektir:

Girdiğim ilk 10 piksellik değer üst margin özelliğini (**margin-top**), ikincisi sağ margin özelliğini (**margin-right**), üçüncüsü alt margin değerini (**margin-bottom**), sonuncusu ise sol margin değerini (**margin-left**) temsil ediyor.

Yani tahmin edeceğiniz üzere, margin özellik grubunda tanımlama yaparken girdiğimiz değerler saat yönünde ilerliyor. Bunu ezberlemenizi öneririm, çünkü fark ettiğiniz gibi bu yöntem yazılacak satır sayısını da mümkün olduğunda azaltıyor, ancak dikkatli kullanılması gerekiyor.

Margin, üstteki örneklerdeki gibi mesafe verme amaçlı pozitif değer alabilirken, gerekli durumlarda negatif değer alarak daraltma veya içeri kaydırma işlevini de üstlenebilir:

```
#anakutu {
  margin: -10px;
}
```

Burada, margin özellik grubuna bu kez tek değer girmiş olduğuma dikkatinizi çekmek istiyorum. Bu benim #anakutu'ya uyguladığım margin değerinin 4 yöne de aynı olacağını belirtme amaçlı bir kısayoldur.

Birkaç bilgi daha vermek istiyorum:

- Margin değerleri her zaman değer belirteci ile kullanılmak zorunda değil. Yani eğer biz sol margin değerine 10 girip hangi uzunluk değeri olduğunu belirtmezsek, tarayıcı bunun bir piksel değeri olduğunu algılayıp o şekilde gösterecektir; örn:

```
#anakutu {
    margin: 10 5 10 5; /* üstten 10, sağdan 5, alttan 10 ve soldan 5
    piksel boşluk */
}
```

- Eğer boşluk verilecek element genişlik ya da yükseklik değerine sahipse (özellikle bu piksel cinsindense) margin için uyguladığımız pozitif değerler ögenin genişlik veya yükseklik değerine ekleme yaparlar. Aynı durumun tersi negatif margin değerleri için geçerlidir.

Bunu aklınızda tutun, çünkü yeri geldiğinde tasarımınızı CSS olarak uygularken sıkıntiya yol açabilir (fazladan genişlik/yükseklik problemi), sorunlu bir görüntülemeye sebep olabilir.

Bunu bir örnekle ifade edeyim:

```
#anakutu {
    width:250px;
    margin-left: 10px;
    margin-right: 10px;
}
```

Burada aslında #anakutu selektörünün uygulandığı elementimiz aslında 250 piksel genişlikle değil, sol ve sağa eklenen boşluklarla 270 piksellik genişlik etkisine sahip olacaktır.

Dolayısıyla anakutuya dış boşluk uygulamak istiyorsak burada yapmamız gereken şöyledir:

```
#anakutu {
    width:230px;
    margin-left: 10px;
    margin-right: 10px;
}
```

Fazla gelen sol ve sağ dış boşluk oranlarını toplayıp, #anakutu genişlik değerinden çıkararak, anakutuya aslında 250 piksel değerine ulaştırmış olduk.

Margin özelliği yalnızca kutu seviyesi (block level – örn:p, div, form vs) elementlere uygulanabilirler.

Margin Tabanlı Muhtemel Sıkıntılar ve Çözümleri

Tarayıcı standardı stillerde bazı elementler kendi birtakım standart değerleriyle birlikte gelir demistik. İşte bu değerlerden özellikle margin, kayma ve fazladan boşluk yaratabilirler, farklı tarayıcılarda farklı görünümü sebep olabilirler.

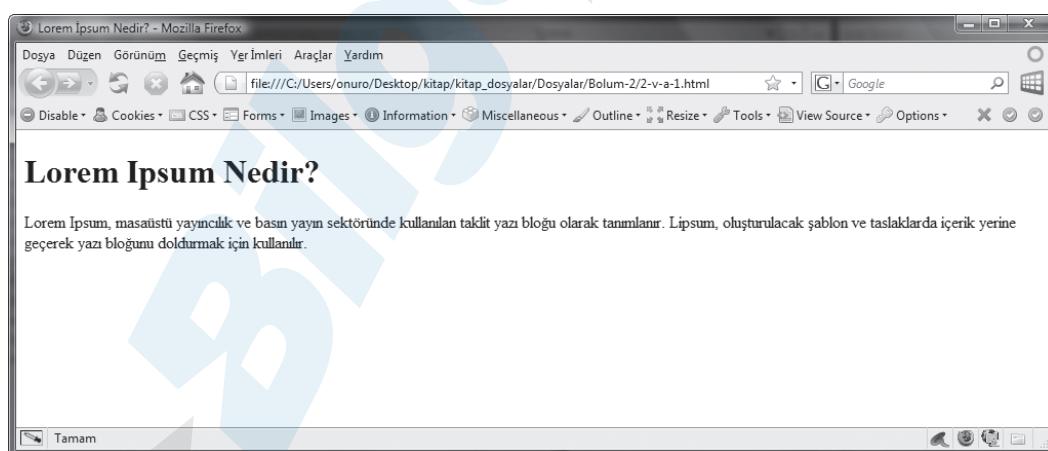
Bunu daha detaylı ifade etmek için bir örnek çalışma uygulamak istiyorum. Öncelikle, içeriğinde anakutu div'inin bulunduğu bir doküman oluşturuyorum:

```
<DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns= "http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html charset=utf-8"/>
<title>Lorem Ipsum Nedir?</title>
<style type="text/css">
/* az sonra buraya css kodlarınıza gireceğiz*/
</style>

</head>
<body>

<div id="anakutu">
<h1> Lorem Ipsum Nedir?</h1>
<p>
Lorem Ipsum, masaüstü yayıncılık ve basın yayın sektöründe kullanılan taklit yazıbloğu olarak tanımlanır. Lipsum, oluşturulacak şablon ve taslaklarda içerik yerine geçerek yazıbloğunu doldurmak için kullanılır.
</p>
</div> <!-- #anakutu kapanışı -->

</body>
</html>
```



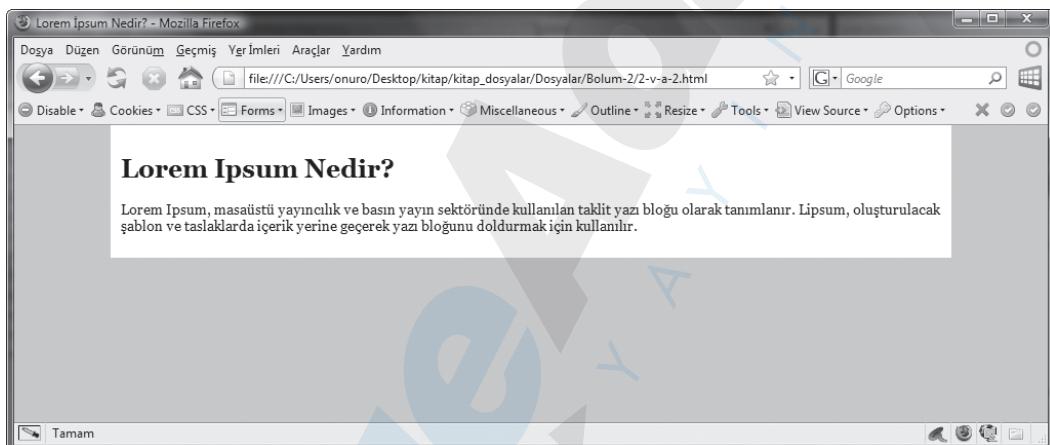
Figür 2.vi.a-1

Şimdi bu dokümanı biraz stillendiriyorum:

```
body {
    margin: 0px; /* dokümanda boşluk bırakma */
    font: 9.5pt Georgia; /* yazıtipi olarak 9.5 punto Georgia belirle */
    background:#ccc; /*arkaplan rengini açık ton gri renk belirle*/
}

#anakutu {
    background:#fff; /*kutu arkaplanını beyaz renkte görüntüle*/
    width:800px; /*800 piksellik genişlik ver*/
    margin:0 auto; /*bir üst elemente göre yatay olarak ortala*/
    padding:10px; /*10 piksellik iç boşluk uygula*/
}
```

Bu uygulamaya göre almamız gereken sonuç, doküman içinde ortalanan bir anakutu ve onun içeriğidir. Firefox tarayıcısında sonuca bakıyoruz:



Figür 2.vi.a-2

Firefox tarayıcısını baz alarak, istediğimiz etkiyi tepedeki boşluk sıkıntısı haricinde elde ettik diyebiliriz.

Aynı uygulamanın Internet Explorer sonucuna da bakacak olursak:



Figür 2.vi.a-3

Firefox'ta çok sıkıntılı olmayan uygulamamız Internet Explorer'da bazı farklarla karşılaşıyor. Aradaki ayırlara göz atalım:

1. Anakutu div'i ortalanmıyor
2. H1 elementinin yazıtipi ve margin oranları Firefox ve Internet Explorer tarafından farklı görüntüleniyor.

Anakutu div'i ortalanmıyor:

Tüm tarayıcılarda da ortalanan kutu özelliğini oluşturmak için bazen örneğimizdeki gibi margin: auto özelliği yeterli gelmeyebilir.

Bu etkinin gerçekleşeceğini emin olmak için ileriki bölümlerde daha detaylı göreceğimiz text-align özelliğinden yardım alacağım. body etiketi için metni text-align özelliğine center vererek ortalayacak olursam dokümandaki tüm elementler ve metinler ortalanacaktır (tabii anakutu içindekiler de). Anakutu içindeki metinleri toplamak içinse onun selektöründe text-align özelliğini tekrar sola almak gerekecek:

```
body {
    margin: 0px; /* dokümanda boşluk bırakma */
    font: 9.5pt Georgia; /* yazıtipi olarak 9.5 punto Georgia belirle */
    background: #ccc; /*arkaplan rengini açık ton gri renk belirle*/
    text-align: center;
}

#anakutu {
    background: #fff; /*kutu arkaplanını beyaz renkte görüntüle*/
    width: 800px; /*800 piksellik genişlik ver*/
    margin: 0 auto; /*bir üst elemente göre yatay olarak ortala*/
    padding: 10px; /*10 piksellik iç boşluk uygula*/
    text-align: left;
}
```

Şimdi, Internet Explorer sonucuna tekrar bakalım:



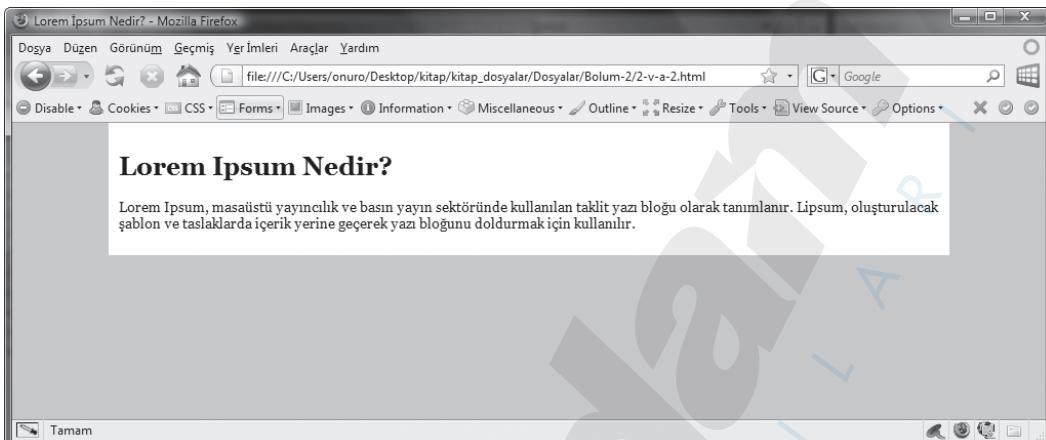
Figür 2.vi.a-4

Neredeyse aynı sonucu yakaladık.

H1 elementinin boşluk ve yazıtipi oranları tarayıcılara göre farklılık gösteriyor:

Bu, oldukça sık karşılaşılan bir sıkıntıdır. Başlık elementleri, tarayıcı standarı stillendirmede kendilerine ait bir yazıtipi punto ve margin, padding boşluk değerlerine sahip olmakla kalmaz, tarayıcıdan tarayıcıya da farklılık gösterebilir.

Keza, son duruma tekrardan bir göz atalım. Öncelikle Firefox sunucumuz:



Figür 2.vi.a-2

Ve Internet Explorer:



Figür 2.vi.a-4

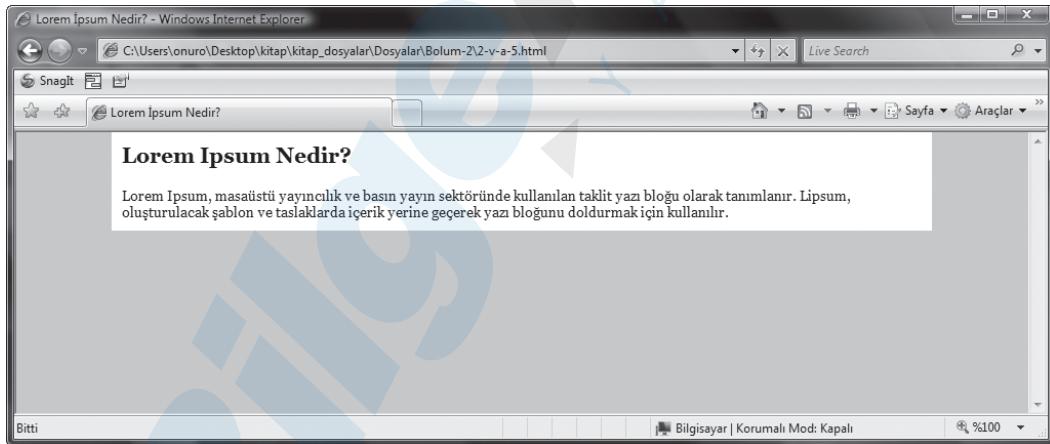
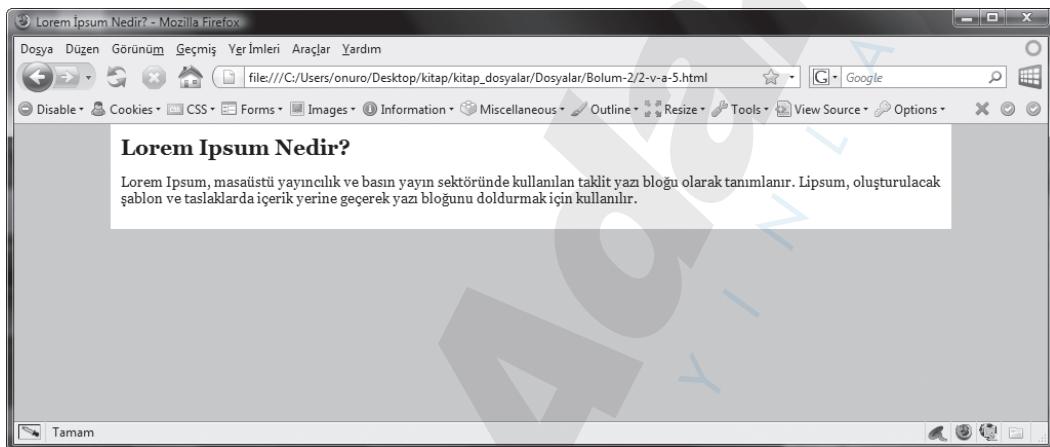
Bu görünüm farkını kaldırmanın yolu, elementin sıkıntı yaratan standart değerlerine CSS yardımıyla biçimlendirme uygulamaktır. Burada sıkıntımız h1 elementinin sahip olduğu standart margin, padding ve punto boyutu değerleridir. O halde margin ve padding'i sıfırlayıp yazıtipine belirgin bir punto oranı verelim:

```
body {
    margin: 0px; /* dokümanda boşluk bırakma */
    font: 9.5pt Georgia; /* yazıtipi olarak 9.5 punto Georgia belirle */
    background: #ccc; /*arkaplan rengini açık ton gri renk belirle*/
    text-align: center;
}
#anakutu {
    background: #fff; /*kutu arkaplanını beyaz renkte görüntüle*/
    width: 800px; /*800 piksellik genişlik ver*/
}
```

```

margin:0 auto; /*bir üst elemente göre yatay olarak ortala*/
padding:10px; /*10 piksellik iç boşluk uygula*/
text-align:left;
}
#anakutu h1 { /*yalnızca anakutu divinin içindeki h1 başlıklar
için*/
margin:0; /*dış boşluğu sıfırla */
padding:0; /*iç boşluğu sıfırla */
font-size: 16pt; /* yazıtının boyutu tüm tarayıcılarda 16 punto
görüntüle */
}

```



İşte bu kadar. İki tarayıcıda da boşluk ve ebat oranları birebir olarak aynı görüntüyü elde ettik.

2.v.b İç Boşluk – Padding

Padding, margin'den biraz farklı olarak CSS ile seçilmiş olan bir HTML elementinin kendi içindeki boşluk oranıdır. Bu, elementin içinde yer alacak öğelerin belli bir mesafeden sonra görüntüleneceği anlamına gelir.

Diğer tüm padding yapı ve özellikleri, uygulanych şekli margin ile neredeyse aynıdır. Tek farkla; padding özelliği negatif değerler ve auto değeri alamaz.

Padding de her yöne ayrı ayrı verilebilir:

- **padding-top:** üst iç boşluk mesafesi;
- **padding-left:** sol iç boşluk mesafesi;
- **padding-right:** sağ iç boşluk mesafesi;
- **padding-bottom:** alt iç boşluk mesafesi;

Bu 4 yöne girebileceğimiz değer uzunlukları:

- Px (piksel) cinsinden uzunluk; **örn:**

```
p {
  padding-left:15px;
}
```

Bu örnekte paragrafin içeriği 15px içерiden başlayacaktır.

- Percent (yüzde) cinsinden uzunluk; **örn:**

```
p {
  padding-top:50%;
}
```

Bu örnekteyse paragrafin içeriği paragraf yükseklik değerinin yarısı oranında aşağıdan başlayacaktır.

Margin'de olduğu gibi, 4 yöne ayrı ayrı özellik açarak biçimleme yapmak yerine, padding özellik grubunu kullanarak uygulama yapmak da mümkündür:

```
#anakutu {
  padding: 10px 10px 10px 10px;
}
```

Burada gene değerler saat yönünde ilerleyerek giriliyor ve tek padding özellik grubu kullanılarak değer girilebiliyor.

Yine margin'de de olduğu gibi, padding için değer belirteci girilmediğinde, tarayıcı bunun piksel cinsinden olduğunu otomatik olarak algılayacaktır.

- Padding özelliği yalnızca kutu seviyesi (block level – örn:p, div, form vb.) elementlere uygunabilirlər
- Padding de margin gibi pozitif değerler girildiğinde öğelerin genişlik ve yüksekliklerine eklenme olarak gelirler.

2.c.vii – Tarayıcılardaki Yorum Farkları

Şu anda piyasada sıkılıkla kullanılan 4 ya da 5 popüler tarayıcı bulunmaktadır.

Her ne kadar İlk bölümde, CSS ile hazırladığımız tasarımların bizi yormadan ortak görünümle tüm tarayıcılarda sorunsuz çalıştığından bahsetmiş olsam da, sıkılıkla problem yaşadığımız bir tarayıcı vardır:

Internet Explorer

Önceki örneklerimizden birinde gördüğümüz, ve sonraki örneklerin bir kısmında da göreceğimiz gibi, Internet Explorer bazı element özelliklerini kendine göre yorumladığı için, dünyanın geri kalan tüm tarayıcılarında sorunsuz görüntülenen birçok stillendirme uygulamamız, Internet Explorer ile muhtemel sorunlara sahip olabilir.

Bu tip durumlarda çalışmalarımızı, Internet kullanıcılarının yarısının tercihi olması nedeniyle ne yazık ki Internet Explorer ile de kontrol ederek görünüm farkı varsa bunları düzelterek yapmamız gerekecek.

Internet Explorer en çok aşağıdaki element özellikleriyle ilgili görünüm farkına neden olur:

- Margin (dış boşluk)
- Padding (iç boşluk)
- Font-size (yazıtipi boyutları)

Dolayısıyla çalışmalarınızda eğer Internet Explorer ile ilgili görünüm sıkıntılara rastlayacak olursanız, ilk olarak yukarıdaki özellikler için standart değerler atayarak Internet Explorer görünüm problemlerini ortadan kaldırabilirsiniz.

Bunun da işe yaramaması durumunda, ileride örneğiyle birlikte göreceğimiz koşullu stil yükleme kullanabiliyor olacağız. Koşullu stil yükleme doküman Internet Explorer ile görüntülendiğinde, ona özel stil özellikleri tanımlamamıza yardımcı olacak.

Internet Explorer ile yaşanan bir diğer kritik problem de transparan *.png uzantılı dosyaların 6.0 versiyonu ile sorunsuz görüntülenmemesi idi. Bu sorun, bu kitaptaki uygulamalarda da test ederken kullanılan ve günümüzde eski versiyonun yerini %60 oranında almış olan 7.0 versiyonu ile çözüldü.

Dolayısıyla, eğer çalışmalarımız Internet Explorer görünümünde sıkıntı yaratırsa onları çözmemiz yeterli olacaktır. Geri kalan tarayıcıların neredeyse tamamı Worldwide Web Konsorsiyumu (W3C) standartlarını en doğru şekilde desteklediği için, Firefox ile sıkıntısız görüntülenen bir çalışma onlarda da düzgün görüntülenecektir.



3

**CSS Yardımıyla
Web Tipografisi /
Yazıtipi
Biçimlendirme**

3 CSS Yardımıyla Web Tipografisi / Yazıtılı Biçimlendirme

- Web İçin Tipografi
- CSS Yardımıyla Şık Görünümlü Yazıtılı Oluşturmak
- Tasarım ile Yazıtılı Görsel Bütünlük Oluşturacak Şekilde Kullanmak
- Daha Estetik Görünüm İçin “Görsel Yerdeğişim” Yöntemi

CSS Yardımıyla Web Tipografisi / Yazıtipi Biçimlendirme

3.a Web İçin Tipografi

Öncelikle tipografi kavramına değinelim:

Tipografi bir grafik tasarım kavramıdır ve dilimize Yunanca'dan girmiştir. Yazıtınızı bir forma, biçimde sokma sanatı olarak tanımlanır. Tasarım kavramının yer aldığı (Web tasarımını da dahil olmak üzere) neredeyse her noktada büyük önem taşıyan bir olgudur.

Tipografi Neden Bu Kadar Önemli

Bu kitabın yayınlandığı dönemde oldukça popüler bir film olan Batman – Dark Knight filminin Web sitesine girdiğinizde, film başlığının google.com.tr logosundaki gibi "Times New Roman" karakterleriyle civil civil karakter renkleriyle yer aldığı gözünüzde canlandırın. Filmin teması ve yapısıyla pek alakalı olmayacağından, değil mi?

Tipografi, tasarımda metinsel içeriğin ziyaretçi veya izleyici tarafından okunacak öğeler haricinde görsel bütünlüğün bir parçası olarak ele alınmasına dayanır.

Yani eğer hazırladığınız projenin konusuyla alakalı olmayan ya da görsel konsept bütünlüğünü bozacak yazıtımı stillendirmelerine girerseniz, tasarımsal kriterlere göre pek başarılı bir sonuç elde etmemiş olacaksınız.

Peki, Web Tasarımında Tipografinin Getirişi Ne Olacak?

Elbette, öncelikle daha akıcı bir içerik görünümü elde edeceksiniz.

Hazırladığınız Web sitesinin yapısıyla uyumlu ve okunaklı yazıtımı büyüklükleri ve renk tonlamaları, metinsel içeriğin görsel hiyerarşisi, siteyi görüntüleyen ziyaretçinin içeriği sıkıntı çekmeden rahat bir şekilde okumasını ve izlemesini sağlayacaktır.

CSS yardımıyla, Web dokümanlarımızdaki yazıtımı ve metinsel içeriğin görünüm düzenlemesini yaparak göze daha akıcı ve okunaklı bir içerik yapısı sunacağız.

3.b CSS Yardımıyla Şık Görünümlü Yazıtımı Oluşturmak

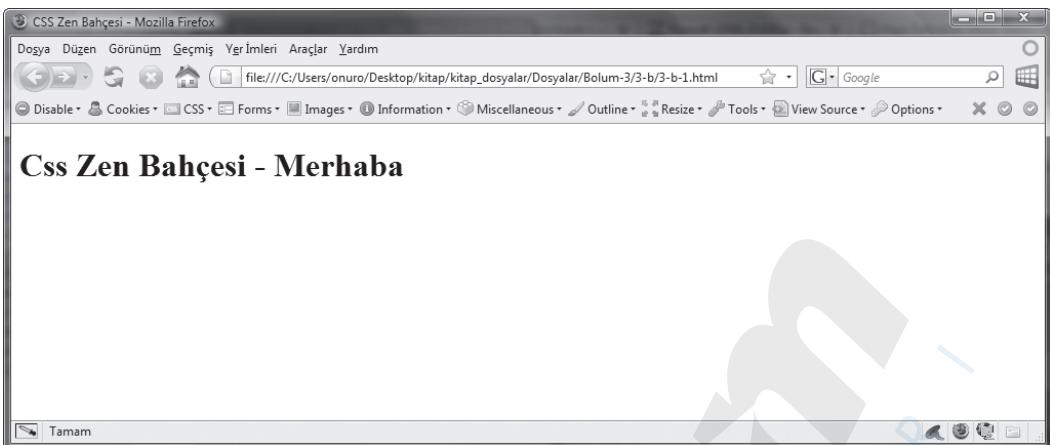
Yazıtımı, tasarımcılar olarak Web tasarımda en sıkıntı duyduğumuz konulardan biridir. HTML dokümanına atayarak tanımladığımız bir yazıtımı, bununla hazırladığımız bir Web sitesini gezen ziyaretçinin bilgisayarında yüklü olmayıp, bu da sitemizin içeriğini ziyaretçinin tarayıcısının kendi standart yazıtımı görünümünde görüntülemesine sebep olarak görsel problemler yaratabilir.

Bu nedenledir ki, elimizdeki 9-10 kadar kısıtlı, her bilgisayarda yüklü olan standart yazıtımı aileleriyle yetinmek durumunda kalıyoruz. Bu yazıtımıyla oluşturduğumuz Web siteleri böylece her bilgisayar tarafından tüm tarayıcılarında sorunsuzca görüntülenebiliyor.

Yazıtımı adedimizdeki limitlere rağmen, CSS içindeki yazıtımı özellikleri (property) yardımıyla bu problemi mümkün olduğunda giderebilecek, yaratıcılığımızı kısıtlayan bu sıkıntıyı bir nebze olsun azaltabileceğiz.

Şimdi, bu bahsettiklerimle ilgili kısa bir ön çalışma yapalım ve oluşturacağımız bir h1 başlığının yazıtımı özelliklerini biçimlemeye çalışalım:

```
<DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  
<html xmlns= "http://www.w3.org/1999/xhtml">
```



Figür 3.b.1

```

<head>
<meta http-equiv="Content-Type" content="text/html charset=utf-8"/>
<title>CSS Zen Bahçesi</title>
<style type="text/css">
/* az sonra buraya css kodlarını gireceğiz*/
</style>

</head>
<body>

<h1> CSS Zen Bahçesi - Merhaba </h1>
</body>
</html>
  
```

Stillendirmemizi henüz uygulamadığımız için, yukarıdaki görüntüde h1 başlığımızın stilsiz, tarayıcı standardı yazıtipiyle (çoğunlukla "Times New Roman") yer aldığığini görüyoruz.

Şimdi, bu başlığa biraz stil atayalım:

```

h1 {
font-family: Verdana, Arial, Helvetica, Tahoma;
font-size:300%;
}
  
```

Burada kullandığımız özellikler ve uyguladığımız biçimlemeler:

1. font-family özelliğiyle selektöre yazıtını değiştireceğimizi belirttik.
2. font-family özelliğine atadığımız çoklu değerlerle tarayıcıya sunları demış olduk:
 - a. Ziyaretçinin bilgisayarında **Verdana** yazıtipi yüklüse h1 başlığı bu yazıtipiyle görüntüle.
 - b. **Verdana** yazıtipi yüklü değilse **Arial** ile görüntüle.
 - c. **Arial** da bulunmuyorsa **Helvetica** ile görüntüle.
 - d. O da yoksa **Tahoma** yazıtipiyle görüntüle.

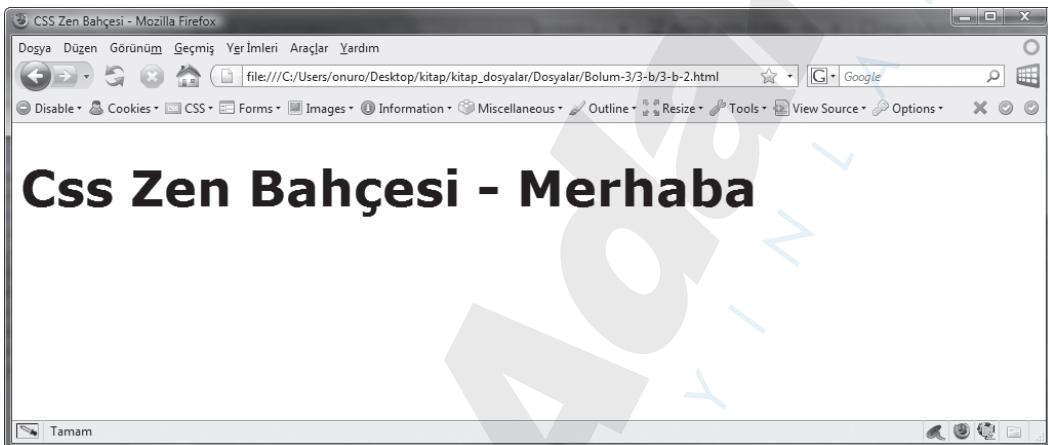
Ziyaretçilerimizin bilgisayarlarında, bu yazıtipileri sistem fontları olduğundan, en az bir tanesi mutlaka bulunacaktır (onlar kendi ellişle özel olarak bu yazıtipilerini silmemişlerse)

3. **Font-size** ile yazıt tipi ebadını değiştireceğimizi tanımladık ve verdiğimiz değerle yazıtının ebadını, sahip olacağım standart ebadından 4 kat daha büyük görüntülemek istediğimi belirtti.

font-size özelliğine atayabileceğimiz değerler yalnızca örnekteki gibi yüzde "%" ile sınırlı değildir.

Yazıt tipi ebat değeri girerken piksel "px" ve punto "pt" ve % gibi bağılı bir değer olan Emphasis "em" oranları girerek de yazıt tipine ait net boyut tanımlamaları yapabiliriz.

Örnekte yaptığımız değişikliklerin sonucuna tarayıcıda bakalım:



Figür 3.b.2

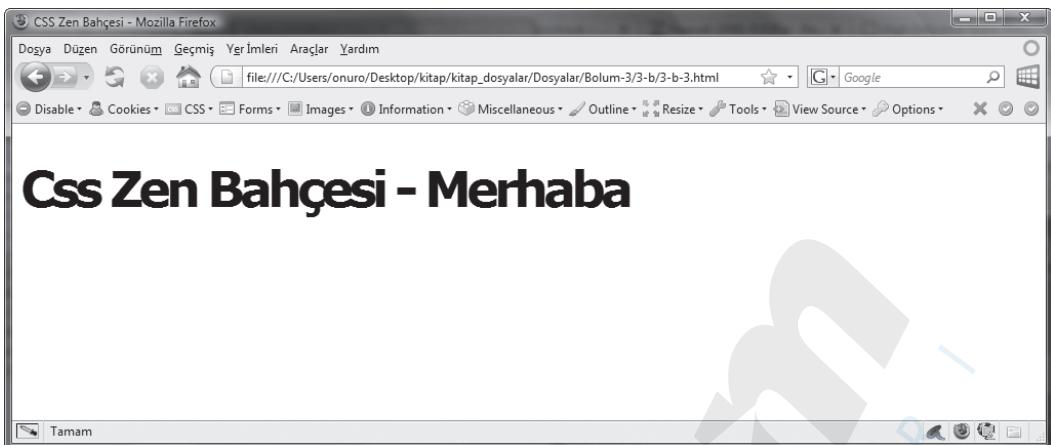
Böylece dokümanımızdaki h1 başlıklarımızın görünümüne standart görünümden farklı bir biçim kazandırılmış olduk.

Şimdi, başlığımızın yazıt tipine bir kaç özellik daha ekleyelim:

```
h1 {
  font-family: Verdana, Arial, Helvetica, Tahoma;
  font-size:300%;
  letter-spacing:-5px;
}
```

"**letter-spacing**" özelliğiyle yazıt tipimizin sahip olduğu karakterlerin arasındaki boşluk oranını değiştireceğimizi belirtti ve bu özelliğe atadığımız değerle karakter aralarının **-5px** oranında daralacağını tanımlamış olduk.

letter-spacing özelliği **font-size**'da olduğu gibi, "px", "pt", "%" ve "em" değerlerini alabilir.

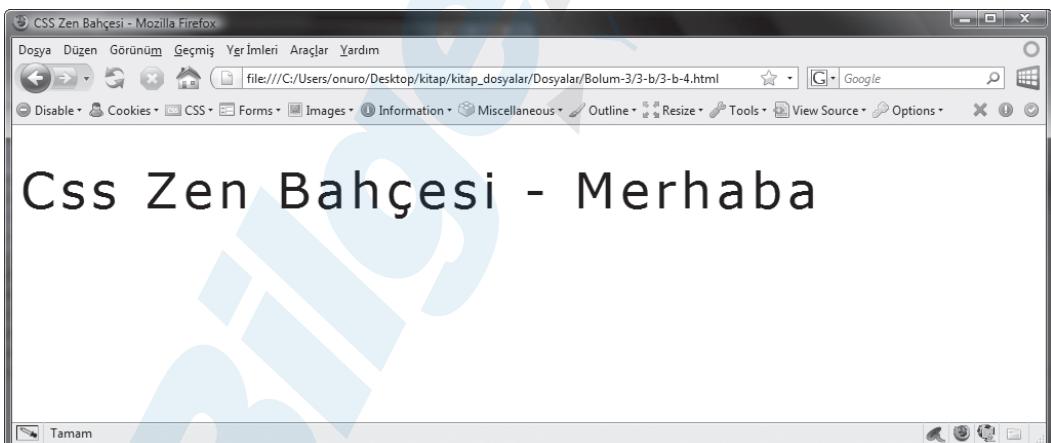


Figür 3.b.3

Şimdiden, yalnızca ufak düzenlemelerle bile CSS yardımıyla Web dokümanlarımıza daha yaratıcı bir açı katabildiğimizi görebiliyoruz.

Birimlemeye devam edelim. Negatif değer verilmiş harf arası boşluğu gördük. Şimdi de pozitif uygulayalım. Yani harflerin arasını kısmak yerine açacağız:

```
h1 {
    font-family: Verdana, Arial, Helvetica, Tahoma;
    font-size: 300%;
    letter-spacing: 5px;
    font-weight: normal;
}
```



Figür 3.b.4

Burada harflerin aralarının açılması haricinde bir önceki örneklerde oranla bacakları daha ince bir yazıtipi görüyoruz. Bunu `font-weight` özelliğine `normal` değeri vererek sağladık. Böyle yapmamızın sebebi, bazı metinler (`h1`, `h2` gibi başlıklar) tarayıcı standardı stillendirmede büyük puntolarla görüntülenmelerinin haricinde kalın olarak tarińırlar. Bunu bazen değiştirme ihtiyacı duyabiliriz.

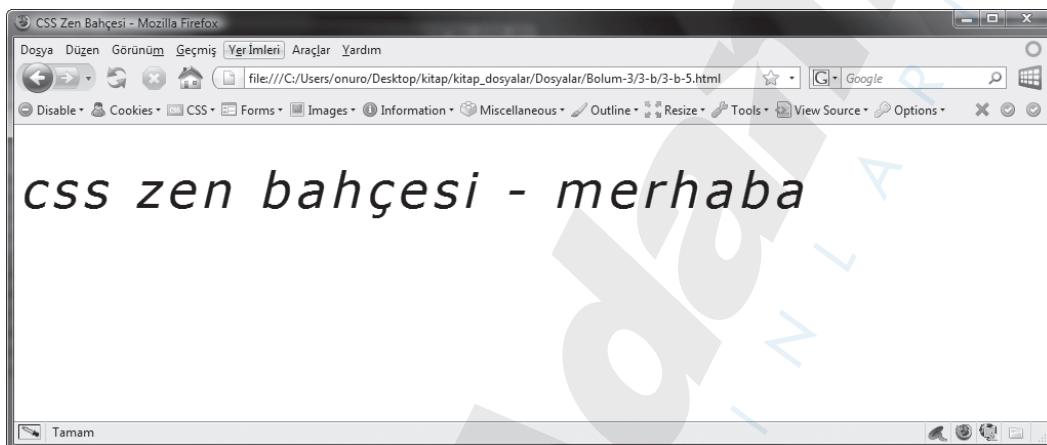
Veya normal bir metin içeriğinin belli bir kısmını kalın göstermek isteyebiliriz. Bu durumda da, `font-weight` özelliğine `bold` değerini atıyor olacağız.

İki stillendirmeye daha dejinmek istiyorum. Şimdi, başlığımızın yukarıdaki özelliklere ek olarak italic ve tamamı küçük harflerle görüntülenmesini sağlayalıım:

```

h1 {
  font-family: Verdana, Arial, Helvetica, Tahoma;
  font-size: 300%;
  letter-spacing: 5px;
  font-weight: normal;
  font-style: italic;
  text-transform: lowercase;
}

```



Figür 3.b.5

Oldukça etkileyici, değil mi? `text-transform` özelliğine uppercase değerini atamış olsaydık, bu kez tüm karakterler büyük harfle görünecekti.

`text-transform` özelliğe benzer bir etkiye sahip olan `font-variant` özelliğinden de bahsetmek istiyorum:

```

h1 {
  font-family: Verdana, Arial, Helvetica, Tahoma;
  font-size: 300%;
  letter-spacing: 1px;
  font-weight: normal;
  font-style: italic;
  font-variant: small-caps;
}

```

Özellikle farklı platformlardan (pda, cep telefonu) erişen ziyaretçilerin çok büyük yazıtipi boyutlarındırmalarında sorun yaşamamaları için bazı durumlarda `text-transform` yerine `font-variant` özelliğini kullanmamız gerekebilir veya bunu sadece estetik amaçlı kullanabiliriz.

Başlıklar gibi özel metin yapılarını biçimlendirmeyi görmüş olduk. Şimdi de doküman geneli metin blokları, yani içerik yazıtımız üzerinde düzenlemelere geçelim.

Öncelikle, içinde bir metin bloğu bulunan bir paragraf hazırlayalım:

```

<DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns= "http://www.w3.org/1999/xhtml">

```

```

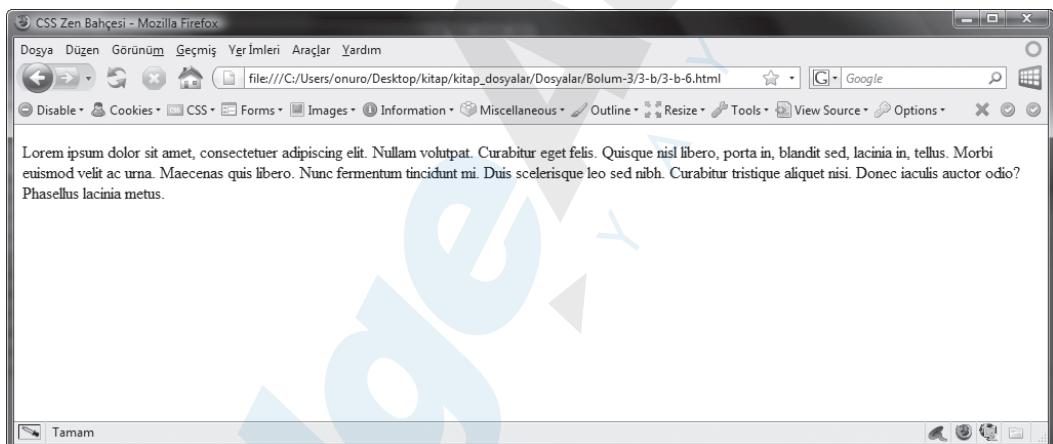
<head>
<meta http-equiv="Content-Type" content="text/html charset=utf-8"/>
<title>CSS Zen Bahçesi</title>
<style type="text/css">
/* az sonra buraya css kodlarınıza gireceğiz*/
</style>

</head>
<body>

<p> Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
Nullam volutpat. Curabitur eget felis. Quisque nisl libero,
porta in, blandit sed, lacinia in, tellus. Morbi euismod velit
ac urna. Maecenas quis libero. Nunc fermentum tincidunt mi. Duis
scelerisque leo sed nibh. Curabitur tristique aliquet nisi. Donec
iaculis auctor odio? Phasellus lacinia metus. </p>

</body>
</html>

```



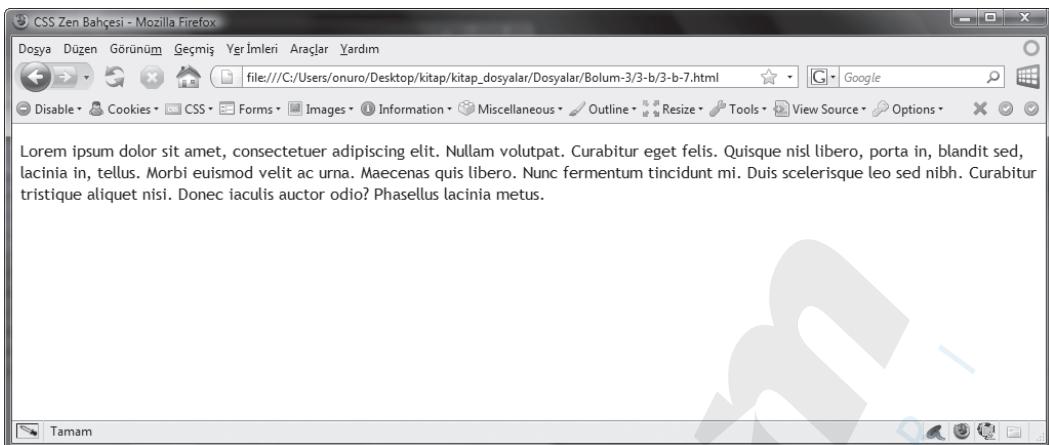
Figür 3.b.6

Paragraf içindeki metin bloğumuzun okunaklılığını artırmak için görünümü üzerinde düzenlemeyeceğiz. Paragrafi düzenlemek için `<p>` etiketi için bir selektör açıyoruz ve biçimliyoruz:

```

p {
font-family: "Trebuchet MS";
line-height: 130%;
}

```



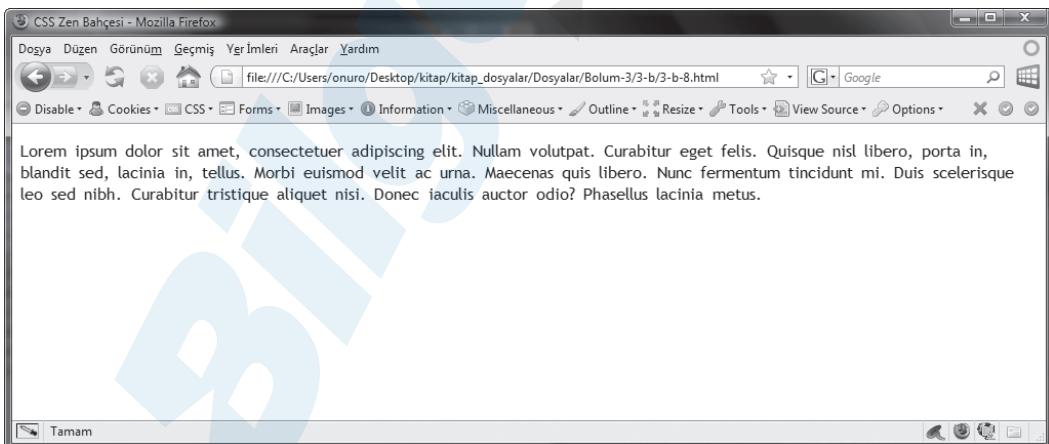
Figür 3.b.7

Yaptığımız biçimlemeyle, paragraf içindeki metin bloğunun fontunu bir sistem yazıtipi olan “**Trebuchet MS**” olarak belirledik ve satır yüksekliğini artırarak, okunaklılığını iyileştirmeye çalıştık.

Bunu yapmak için `line-height` özelliğine sahip olduğu standart değerden biraz daha fazlasını verdik. Diğer metin düzenleme özelliklerinde olduğu gibi, `line-height` özelliği de “`px`”, “`pt`”, “`%`” ve “`em`” değerlerini alabilir.

Paragrafımızdaki kelimelerin aralarını biraz açarak, okunaklılığını daha da iyileştirelim. Bunu uygulayabilmek için, `word-spacing` özelliğine ihtiyacımız olacak:

```
p {
    font-family: "Trebuchet MS";
    line-height: 130%;
    word-spacing: .2em;
}
```

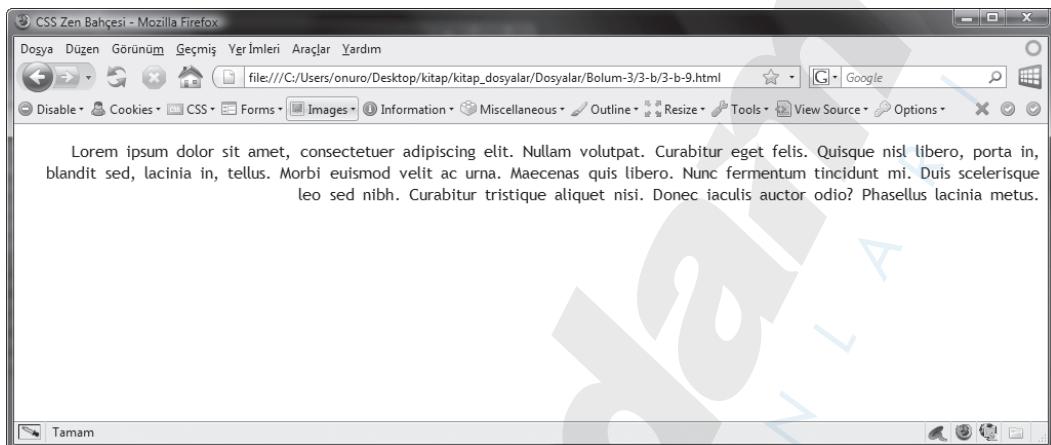


Figür 3.b.8

Burada kullandığımız emphasis “`em`” değeriyle, `word-spacing` özelliğine bağlı bir değer atayarak tüm tarayıcılarda oranlı ve düzgün bir boşluk oranı görüntülenmesini sağladık. Diğer metin düzenleme özelliklerinde olduğu gibi, `word-spacing` özelliği de “`px`”, “`pt`”, “`%`” ve “`em`” değerlerini alabilir.

Şimdi de, metin hizalama yöntemlerini görelim. Standartta sola yaslı olan metin bloğumuzu, sağa yaslanacak şekilde düzenleyelim.

```
p {
    font-family: "Trebuchet MS";
    line-height: 130%;
    word-spacing: .2em;
    text-align: right;
}
```

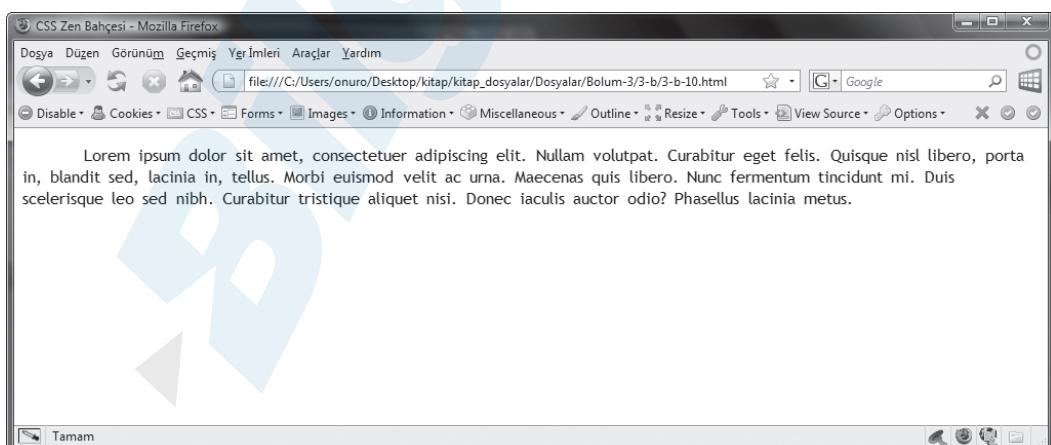


Figür 3.b.9

`text-align: right` haricinde `left`, `center` ve `justified` değerlerini alabilir.

Şimdi de son olarak paragraf başlangıç metnini biraz içeri alıp metni tekrar sola yaslayalım:

```
p {
    line-height: 130%;
    word-spacing: .2em;
    text-align: left;
    text-indent: 45pt;
}
```



Figür 3.b.10

Son eklediğimiz `text-indent` özelliyle, metnimizin ilk satırının 45 punto “`pt`” içерiden başlayacağını belirttik. Eksi değer vererek daha geriden de başlatabilirdik. Diğer metin düzenleme özelliklerinde olduğu gibi, `text-indent` özelliği de “`px`”, “`pt`”, “`%`” ve “`em`” değerlerini alabilir.

Buraya kadar olan alıştırmalarımızda, CSS yardımıyla minimal seviyedeki düzenlemelerle bile oldukça estetik ve sık içerik düzenlemesi yapabileceğimizi görmüş olduk.

Biçimleme ekleyerek veya değiştirerek varyasyonları artırmak oldukça mümkün. Bu konuda kendinizi geliştirmek için selektörlerle atadığınız özellikleri ve değer farklarını uygulayarak bol bol denemeler yapın.

3.c – Tasarım ile Yazıtipini Görsel Bütünlük Oluşturacak Şekilde Kullanmak

Bir önceki bölümde, sistem yazıtiplerini kullanarak onları daha albenili hale getirmeyi, ziyaretçilere elimizdeki kısıtlı imkanlarla dahi daha okunaklı, daha stil sahibi içerik sunumunu görmüştük.

Şimdi, bir adım daha ileri gidip yazıtipi ve grafik kombinasyonunu CSS ile kullanarak Web dokümanlarımızı daha da çekici hale getirmek üzerinde duracağız.

Bazı durumlarda, yaptığımız tasarımların özellikle Navigasyon (site dolaşım menüsü) ve başlık içeriğinde kullandığımız yazıtipinin daha çekici olması için sistem fontları dışında, yalnızca bizim bilgisayarımızda bulunan ve tasarım editörü programımızda daha etkili görünmesi için uyguladığımız efektlerle yer aldığı farz edelim.

Normal şartlarda grafik olarak kaydedilmesi gereken bu metin parçaları, ancak eski yöntem tablolu tasarım ve metin yerine imaj yerleştirmeye mümkün olabilir (**bu, imajlar kapatıldığında ziaretçinin o metin içeren grafik yerine hiçbir şey görmemesine, belki sitenin önemli kısımlını bu nedenle okuyamamasına neden olacak**).

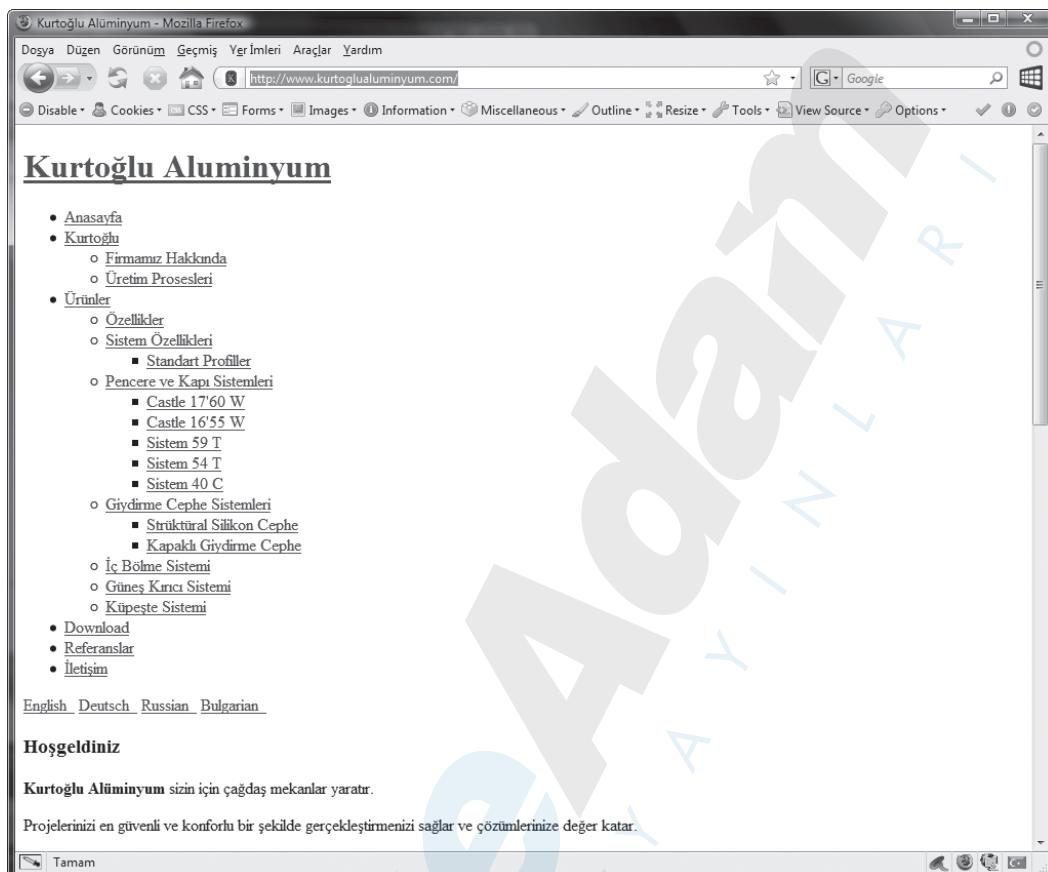
Dolayısıyla grafik haline dönüştürülmüş metin içeriği doğrudan (img etiketi ile) imaj olarak yerlesitmek pek parlak bir çözüm değildir.

İşte burada, CSS'in bize sınırsız yardımcı olacak. CSS ile hazırlanmış başarılı birkaç Web sitesi örneğine bakalım:



Görüldüğü gibi, CSS ile hazırlanmış bu Web sitesinde içeriğin önemli bir kısmı sistem fontlarıyla görüntülenip stillenmişken, özellikle site dolaşım menüsü metin içermesine rağmen grafik bir arayüzden oluşuyor.

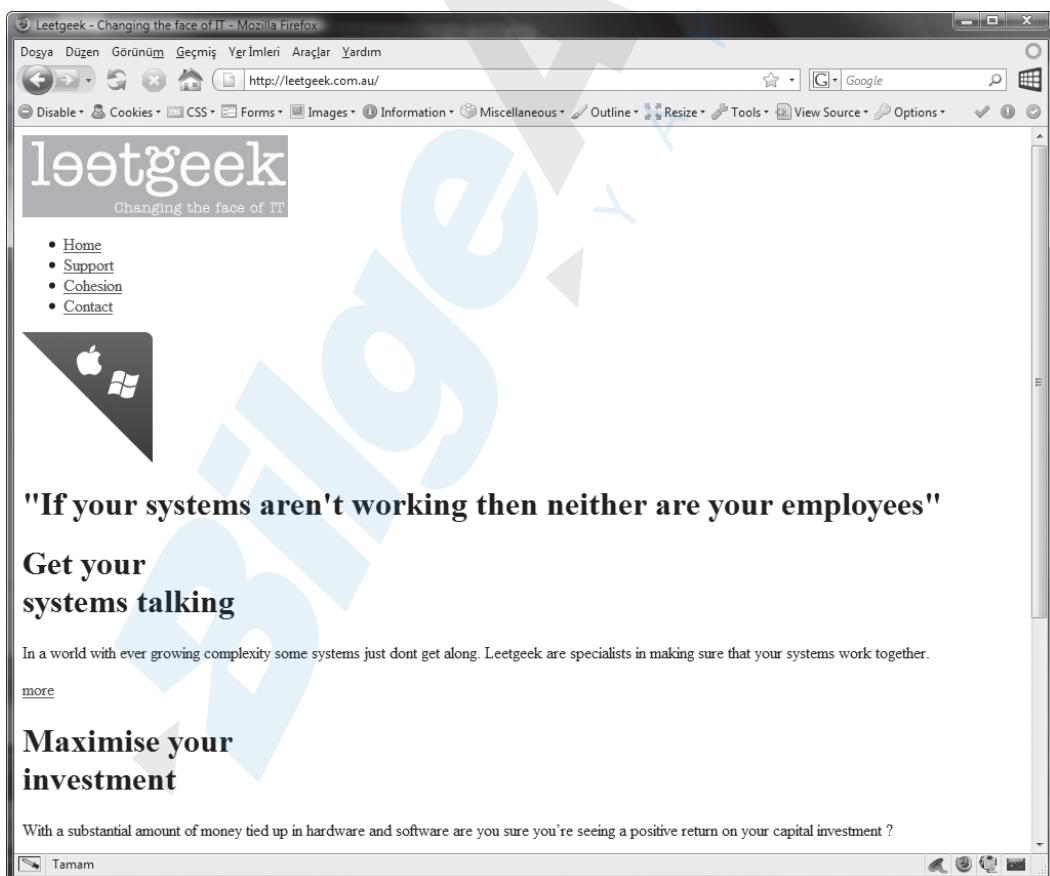
Sitenin CSS'ini kapatacak olursak, yani navigasyon menüsünü imajsız görüntülemek istersek;



Stil veya imajlar kapatıldığında dahi navigasyon menüsünün halen orada olduğunu görürüz.

CSS ile gerekli durumlarda metin yapısı yerine grafik gösterim şansı elde edeceğiz.

Bu tip teknikler uygulanarak hazırlanmış oldukça etkileyici bir başka çalışmaya göz atalım:



Toparlayacak olursak; CSS bize hazırladığımız tasarımları herkesin ulaşabileceğii bir şekilde ve tasarım editöründeki haliyle neredeyse birebir olarak hayatı geçirmemizi sağlıyor.

3.d – Daha Estetik Görünüm İçin “Görsel Yerdeğişim” Yöntemi

Bir önceki bölümde, sistem yazıtiplerini kullanarak onları daha albenili hale getirmeyi, ziyaretçilere elimizdeki kısıtlı imkanlarla dahi daha okunaklı, daha stil sahibi içerik sunumunu ve sınırları daha da genişlettigimizde neler yapabildiğimize dair birkaç örnek görmüştük.

Şimdi biraz daha detaya inip, ziyaretçilere metinlerimin önem taşıyan kısımlarında daha sık sunum ve daha estetik görünüm nasıl kazandırabiliriz, bunun üzerine eğileceğiz.

Ziyaretçiye stilli görünümde estetik ve sık bir grafik metin sunarken, stil kapatıldığından veya stil desteklemeyen bir platformdan girildiğinde bu grafik metnin halen okunabilir sistem yazıtipiyle görüntülenen bir metin halinde kalmasını sağlamak için “Image Replacement – Görsel Yerdeğişim” yöntemi kullanacağız.

Görsel yerdeğişim yöntemi, uygulanacak elementin içindeki metnin gizlenerek, yerine aynı metni içeren bir arkaplan grafiği yerleştirme temeline dayanır. Böylece CSS açıkken ziyaretçi estetik ve stil sahibi bir grafik metin görüntülerken, stil kalkacak olursa halen okuyabileceği bir standart HTML metin ile karşılaşacak.

Dolayısıyla bu teknik bize hem sık, hem de erişilebilir bir metin olanağı sunuyor.

Şimdi, bu tekniğin uygulanışına biraz göz atalım. Dokümanımızda, “CSS ile Tasarımcılara Daha Fazla Özgürlük!” içeriğine sahip bir h2 başlığımız var ancak biz bunu Adobe Photoshop ile bazı efektlere ve sistem yazıtipi olmayan, özel stile sahip bir yazıtipiyle grafik olarak hazırladık. Hazırladığımız bu grafik, **535 piksel genişliğe** ve **65 piksel yüksekliğe** sahip bir jpeg dosyası:



Figür 3.d.1

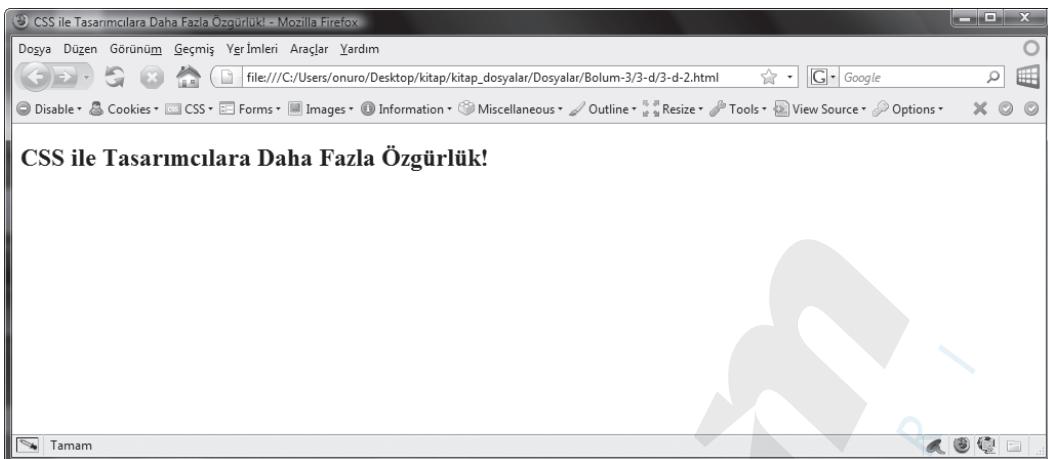
Yapmamız gereken, HTML dokümanı yayınlandığında h2 elementi içindeki metin yerine hazırladığımız bu grafiği göstermek. Bunun için öncelikle jpeg dosyamızın karakteristik özelliklerini h2 elementine atamaya başlayalım:

İlk olarak başlığımızın bulunduğu HTML dokümanımızı hazırlayalım:

```
<DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns= "http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html charset=utf-8"/>
<title>CSS ile Tasarımcılara Daha Fazla Özgürlük!</title>
<style type="text/css">
/* az sonra buraya css kodlarını gireceğiz*/
</style>

</head>
<body>

<h2> CSS ile Tasarımcılara Daha Fazla Özgürlük! </h2>
</body>
</html>
```

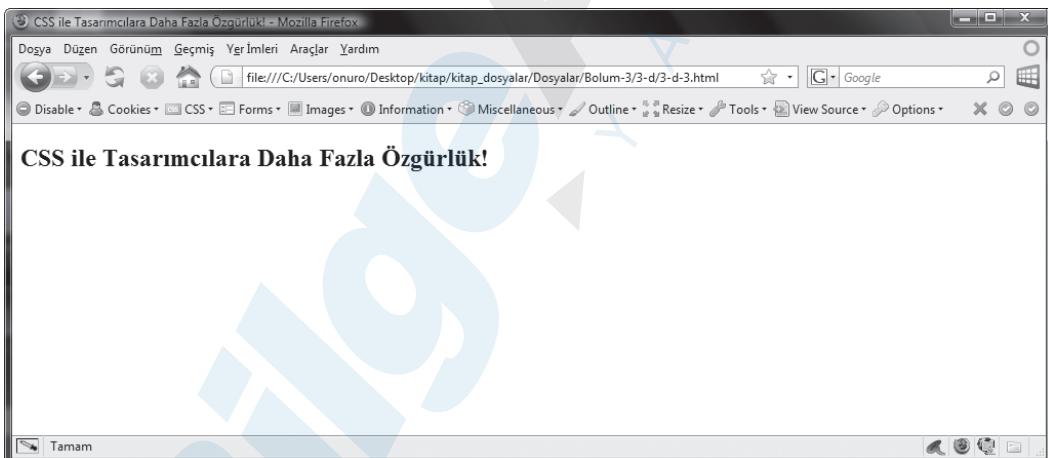


Figür 3.d.2

Stilsiz olarak dokümanımız yukarıdaki gibi görünüyor. Şimdi bu metnin yerini alacak grafiğin bazı özelliklerini h2 etiketine atayalım:

```
h2 {
width: 535px
height: 65px;
}
```

Jpeg dosyamızın sahip olduğu genişlik ve yükseklik özelliklerini h2 elementimize atamış olduk:

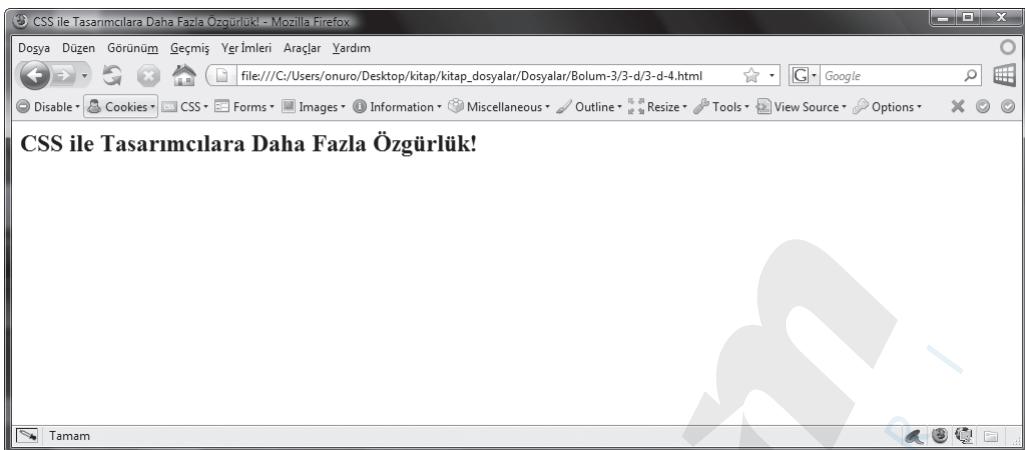


Figür 3.d.3

Görünürde çok bir değişim olmasa da, h2 elementimiz genişlik ve yükseklik cinsinden belirttiğimiz sınırlara indirgenmiş oldu.

Daha önce de belirttiğim gibi, HTML dokümanındaki birçok element, tarayıcı standarı stillendirme ile (stilsiz olarak) görüntülenirken, kendilerine ait bazı varsayılan özelliklere (margin, padding, font boyutu) sahip olurlar.

Bu özelliklerden spesifik olarak margin ve padding, hazırladığımız kompleks bir tasarımın veya grafiklere uyguladığımız tekninin bozulmasına neden olacak kaymala sebep olabilir. Bunu önlemek için margin ve padding değerlerini sıfırlamamız gerekiyor:



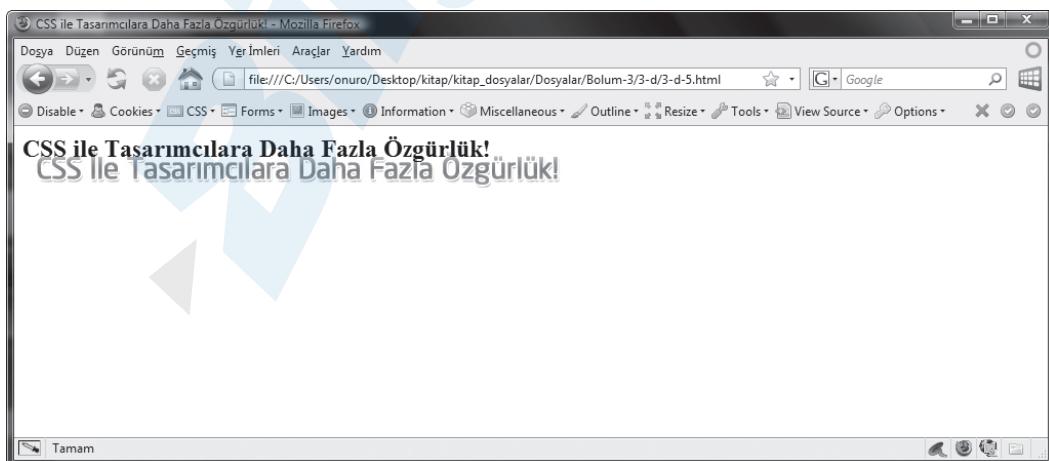
Figür 3.d.4

```
h2 {
    width: 535px;
    height: 65px;
}
```

Böylece, h2 başlığımıza boşluk değerlerini sıfırlayarak ona tüm tarayıcılarda aynı standart fiziksel yapıyı kazandırdık. Artık jpeg dosyamızın kendisini bu elemente bir arkaplan grafiği olarak atayabiliriz. Bu işlem için, “**CSS ile Elementlere Arkaplan Grafiğeri Atamak**” bölümünde daha detaylı olarak göreceğimiz “background” özellik grubuna ihtiyacımız olacak.

Bu özellik grubuna jpeg dosyamızın fiziksel yol değerini atıyoruz ve başlık ebatları içinde tekrar etmemesini sağlıyoruz:

```
h2 {
    width: 254px
    height: 65px;
    margin: 0;
    padding: 0;
    background: url(images/h2baslik.jpg) no-repeat;
}
```



Figür 3.d.5

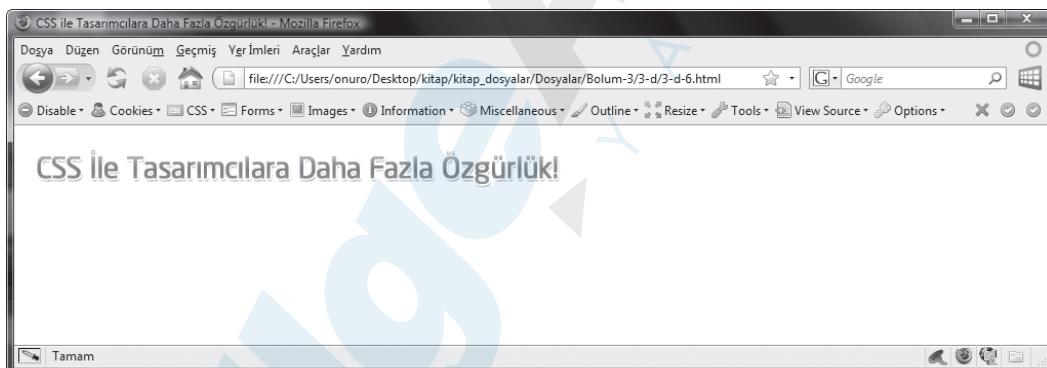
Görüldüğü gibi başlık ile aynı metne sahip grafiğimiz başlık metni arkasına yerleştı. Şimdi yapmamız gereken; arkaplan grafiğimiz ile bu metin içeriğini görüntülerken, varolan h2 başlık metnini küçük bir sihirbazlıkla yok etmeye (aslında gizlemeye) çalışmak.

Bu etki için uygulayabileceğimiz birkaç teknikten iki tanesini göreceğiz. İlk olarak daha önce CSS Yardımıyla Web Tipografisi bölümünden hatırlayacağınız **text-indent** ile gizleme tekniğinden bahsetmek istiyorum.

Bildiğiniz üzere **text-indent** öncelikle ilk satır metinlerinin biraz daha içерiden başlaması için boşluk verilmesine yarıyor. Bu boşluğu elde etmek için pozitif (artı +) değer yerine negatif (eksi -) değer atayabileceğimizden de bahsetmiştim.

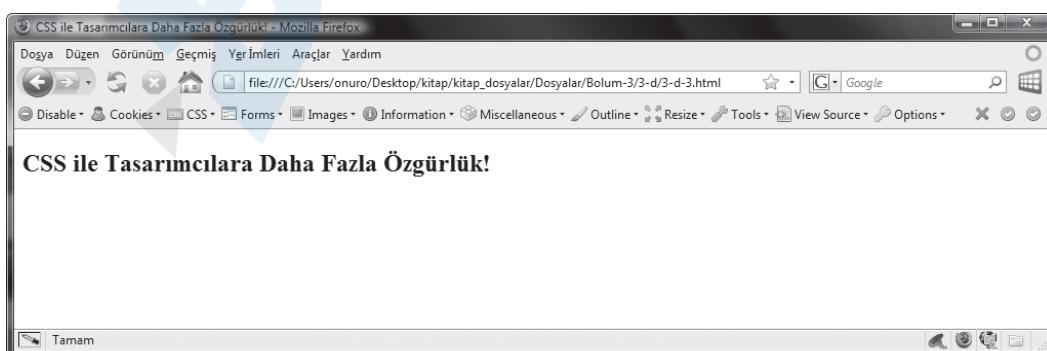
Eğer h2 elementimiz içindeki metnin **text-indent** özelliğine oldukça yüksek bir **negatif** piksel değeri atayacak olursak; tahmin edeceğiniz gibi bu metin aslında bir yere kaybolmayacak, ancak gözden oldukça ırak bir yerde duracaktır:

```
h2 {
    width: 254px
    height: 65px;
    margin: 0;
    padding: 0;
    background: url(images/h2baslik.jpeg) no-repeat;
    text-indent: -9999px;
}
```



Figür 3.d.6

Muhteşem değil mi? Dokümandaki css'i kapatarak uyguladığımız özelliklerin kalkmasını sağlayacak olursak, aslında metnimizin halen orada bulunduğuunu görürüz:



Figür 3.d.7

Görsel Yerdeğşim yönteminin en popüler uygulama tekniğini görmüş olduk. Ancak `text-indent` üzerine negatif değer vererek metni gizlemek ne kadar popüler olsa da, çok nadiren genel tasarım yerleşimi (layout) hazırlarken kayma ve bozulmalara neden olabiliyor.

Böyle ufak bir riski tamamen ortadan kaldırabilecek, biraz daha gelişmiş ancak kesin çözüm niteliği taşıyan bir diğer yöntemeye göz atalım:

Alt Element İçerigini Gizleyerek “Görsel Yerdeğşim”

CSS özellikleri içinde, sahip olduğu değerlerle oldukça güçlü ve sayısız farklı etkiyi oluşturmada kullanabileceğimiz bir özellik var: `display`.

HTML dokümanları içindeki elementlerin `block` (block) veya satır içi (inline) özelliklerini birbirlerine çevirebilen, liste öğelerini (`ul`, `li`) tablo kutuları gibi gösterebilen ve bunlar gibi birçok etkileyici güçe sahip `display` özelliğinin gizleme (`none`) değerini kullanacağımız.

Öncelikle dokümanımızdaki `h2` elementimiz içindeki metni satır içi (inline) etkiye sahip bir element olan `span` ile kapsyalım:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns= "http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html charset=utf-8"/>
<title>CSS ile Tasarımcılara Daha Fazla Özgürlük!</title>
<style type="text/css">
/* az sonra buraya css kodlarımızı gireceğiz*/
</style>

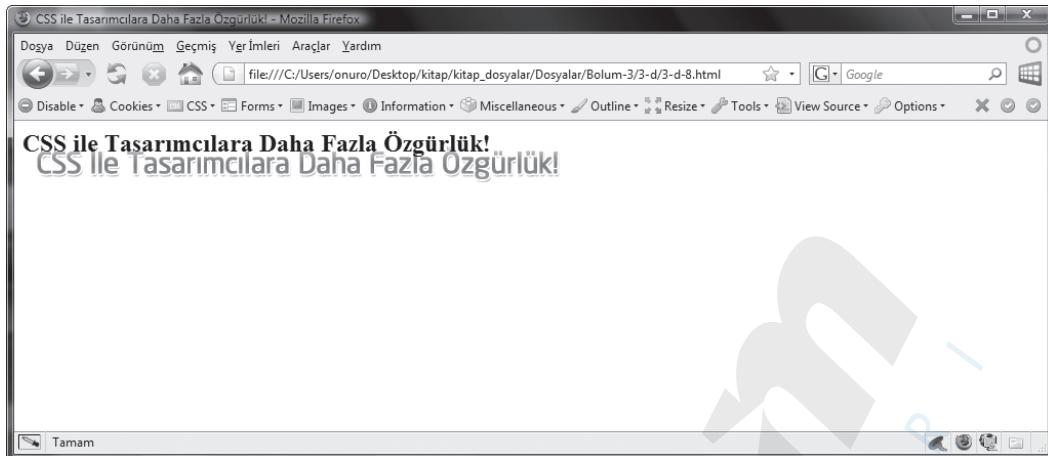
</head>
<body>

<h2> <span>CSS ile Tasarımcılara Daha Fazla Özgürlük! </span> </h2>

</body>
</html>
```

Teknik olarak önceki örnekteki `text-indent` özelliğine negatif değer vererek `span` içinde bulunan metni gizleyebiliriz ancak amacımız bu değil. Amacımız `span` içindeki metni gizlemek (göstermemek). `text-indent` hariç olmak üzere `h2` elementimize tüm özelliklerini bir önceki örnekteki gibi atıyoruz:

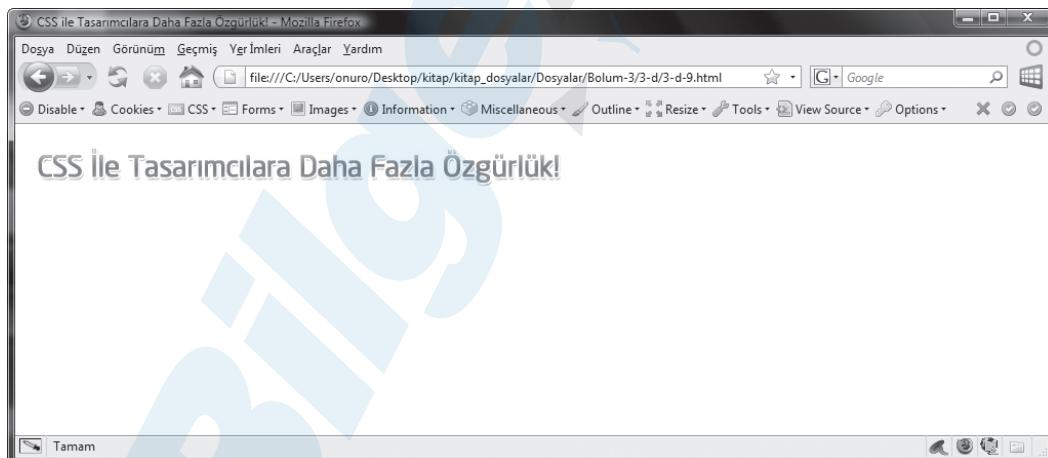
```
h2 {
width: 535px;
height: 65px;
margin: 0;
padding: 0;
background: url(images/h2baslik.jpeg) no-repeat;
}
```



Figür 3.d.8

Şu an arkasında grafik bulunan başlık metni görünümümüze geri dönmüş oldu. Şimdi sıra başlığımızdaki span etiketini ortadan kaldırılmaya geldi. Bu iş için evvelce değiştirdiğim child selektör seçicisini kullanacağız:

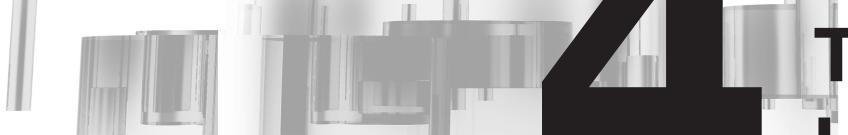
```
h2 {
    width: 254px;
    height: 65px;
    margin: 0;
    padding: 0;
    background: url(images/h2baslik.jpeg) no-repeat;
}
```



Figür 3.d.9

display özelliğinin none özelliğiyle h2 elementi içinde bulunan tüm span etiketlerini kapatarak stilli görünümde bu elementi ve içeriğini gizlemiş olduk. Böylece sıfır risk ile elde ettiğimiz etkiye ulaştık.

display:none teknigi bize daha birçok farklı noktada görüntülemek istemediğimiz öğeleri kaldırmak amacıyla destek olacaktır.



4

CSS ile Tasarımda Renk Uygulama

4 CSS ile Tasarımda Renk Uygulama

- Web Tasarımda Renk ve Kontrastın Önemi
- CSS içindeki Renk Tanımlamaları
- Web Dokümanları İçindeki Elementleri CSS ile Renklendirmek
- CSS Renklendirme ile Derinlik Oluşturmak
- Kontrast Yönetimi

CSS ile Tasarımda Renk Uygulama

4.a Web Tasarımında Renk ve Kontrastın Önemi

Renk, tüm tasarım ile alakalı sektörlerin ana konularından biridir. Amatörler tarafından çoğunlukla göz ardı edilen, ancak hazırladığımız tasarımlarda hayatı önem taşıyan renk unsuru, çalışmamızın profesyonel veya son derece amatör görünmesi hususunda belirleyici rol oynar.

Doğru renk seçimi ve kontrast uygulamaları tasarımlarımızın ziyaretçiler tarafından daha fazla kabul görmesini, daha fazla kitleye mesajımızı iletebilmemizi, işlediğimiz konuyu veya temayı daha iyi ifade edebilmemizi sağlayacak.

Bu kitapta size en doğru renk yaklaşımlarını ve renk teorilerini anlatmayacağım. Bununla ilgili bilgiyi grafik ve Web tasarımla ilgili renklere dayalı kaynaklardan edinebilirsiniz. Ancak renkle ilgili fikir ve uygulamalarımı en iyi şekilde CSS yardımıyla nasıl hayata geçiririz, bunun üzerine deagineceğim.

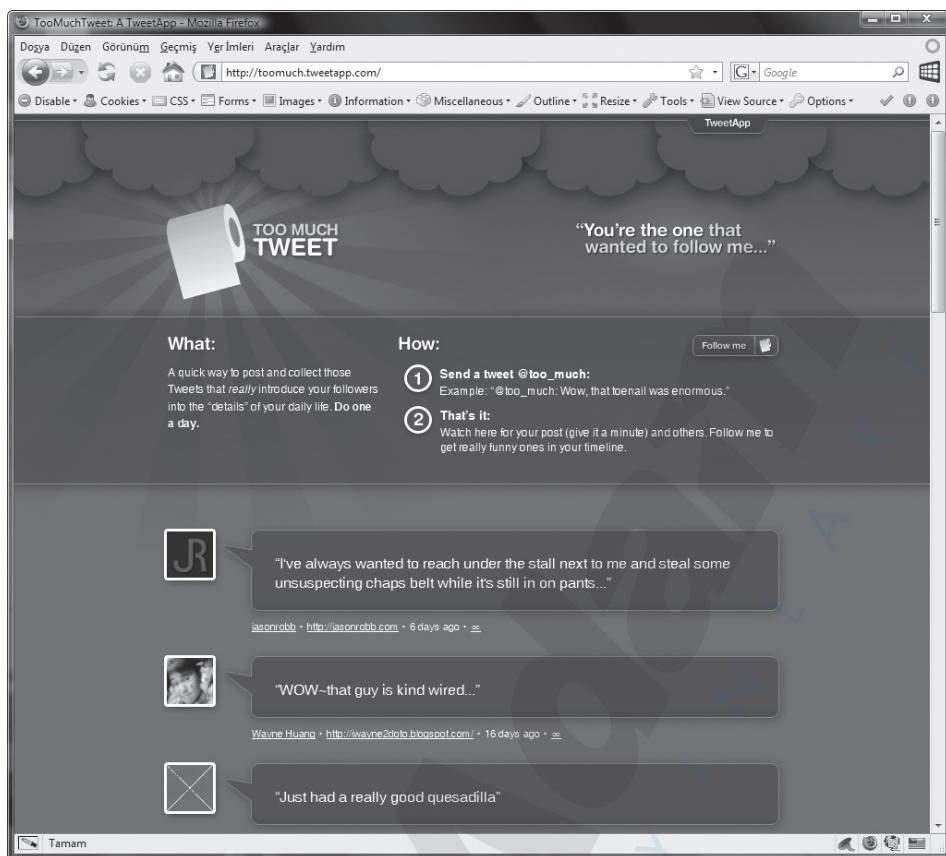
Öncelikle, bu konuda oldukça başarılı bir çalışmayı size önekleyerek, doğru renk kullanımı uygulanmış ve CSS ile hayata geçirilmiş bir Web sitesinden ekranlar sunmak istiyorum:



Kullanıcılarının anlık durumlarını gösteren bir online uygulama olan twitter (www.twitter.com) için özel uygulamalar geliştiren bir servisin Web sitesi olan tweetapp.com, fistık yeşili ve sarı ana tonları kullanılarak hazırlanmıştır.

Renk tonlarının geçisi ve kontrastın (renk karşılığı) dengeli bir şekilde uygulanışı sitenin gözü yormadan oldukça sık bir şekilde görüntülenmesini sağlıyor.

Bunların yanında uygulanan grafik tabanlı efektler bakacak olursak; Photoshop konusunda bilgi sahibi hiçbirinizin "ben bunu yapamam" diyeceğinizi sanmıyorum. Aksine bu siteyi başarılı yapan etmenler en gelişmiş görsel efektleri kullanması değil, doğru yerde doğru görsel elementlerin kullanımıdır:



Sitenin alt bölümlerinden istatistik sayfalarına göz attığımızda, bu kez aynı konseptin farklı renk yaklaşımına şahit oluyoruz.



Help kısmına göz atacak olursak da bu sefer gene aynı görsel yapının mavi tonlarla kullanılışını görüyoruz.

3 ana renk tonu da site genelinde uyum oluşturacak şekilde dağıtılmış ve metin yazıtipileri, grafik ve içerik renklerinin dağılımı, kontrastın kullanımı ve detaylara verilen önem bütünlük oluşturarak sitenin hem kolay gezilebilen, hem de görsel olarak son derece çekici bir yapıda sunulmasını sağlıyor.

Burada sadece site tasarımcısının görsel kabiliyetinden değil, sitenin CSS başarısından da söz ediyoruz. Oluşturulan tasarımın grafik hali ile doküman haline dökülmüş versiyonu CSS yardımıyla birebir şekilde hayatı geçirilmiştir.

Bilindiği üzere, basılı materyaller için renk çalışmaları yapmak ile Web için renk çalışmak farklı kategorilerdir. Web tasarımında ekran/monitör faktörü devreye girdiği, insanların yaş oranlarına göre bile ekranı okuyabilme sürelerindeki kapasite değiştiği için, renk tercihleri ve uygulamada tonların hem etkileyici olması hem de gözü yormaması hususlarında mümkün olduğunda dikkatli olmamız gerekecek.

CSS bize hazırladığımız görsel çalışmalarındaki renkleri en doğru karşılıklarıyla sunmada geniş bir uygulama yelpazesi sunuyor olacak.

4.b CSS İçindeki Renk Tanımlamaları

Web dokümanlarımız üzerine renk uygulamaları yaparken, CSS bize tasarımımızda kullanmış olduğumuz renkleri en doğru şekilde tutturabilmek için tamamı rgb renk paletine bağlı olan **3** farklı renk değer kategorisi girme olanağı sunuyor.

1. Renk Tanımları

Renk tanımları, popüler renklerin İngilizce adlarından oluşur. Bilinen Internet tarayıcıları ve Web standartları konsorsiyumu (W3C-www.w3c.org) bu renk tanımlarından **16** tanesine kesin destek sunar.

Bunlar aşağıdaki şekildedir:

- **Aqua:** deniz mavisi
- **Black:** siyah
- **Fuchsia:** fuşya
- **Gray:** gri
- **Green:** yeşil
- **Lime:** limon sarısı
- **Maroon:** bordo
- **Navy:** koyu lacivert
- **Olive:** zeytin yeşili
- **Purple:** mor
- **Red:** kırmızı
- **Silver:** gümüş
- **Teal:** tuğla
- **White:** beyaz
- **Yellow:** standart sarı

Tahmin edeceğiniz üzere bu kısıtlı renk adediyle hazırladığımız komplike renk paletine sahip bir tasarım çalışmalarımızı hayatı geçirmemiz pek mümkün olmayacağındır. Ancak bu tonları kullanarak bazı temel öğelerin renk özelliklerini tanımlayabiliriz.

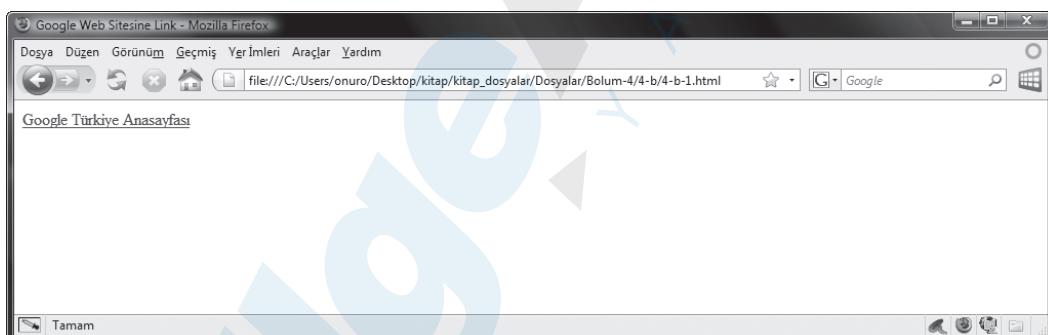
Şimdi dilerseniz İlk olarak bir doküman oluşturalım ve içerik olarak tıklandığında Google'a gönderme işlemini yapan bir link oluşturalım, daha sonra bu link rengi üzerinde değişiklik sağlayacağız:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns= "http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html charset=utf-8"/>
<title>Google Web Sitesine Link</title>
<style type="text/css">
/* az sonra buraya css kodlarını gireceğiz*/
</style>

</head>
<body>

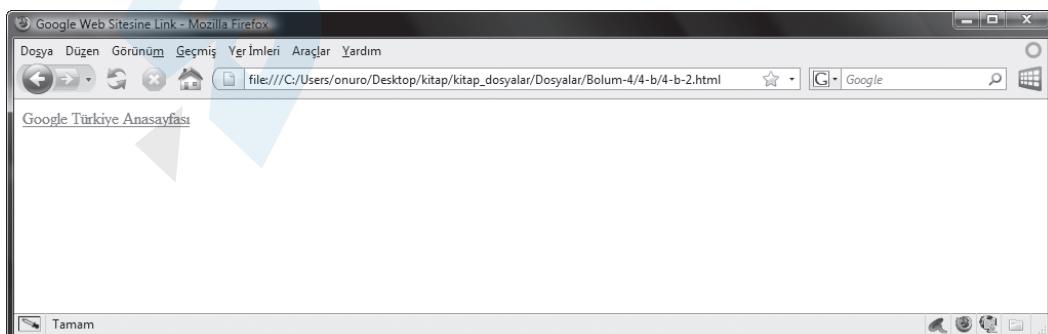
<a href="http://www.google.com.tr">Google Türkiye Anasayfası</a>

</body>
</html>
```



Figür 4.b.1

Şimdi CSS ile bu linkin metin rengi üzerinde “renk tanımı” yaparak link içeriğindeki metnin kırmızı görünmesini sağlayalım:



Figür 4.b.2

color özelliğine atadığımız red renk tanımı ile a etiketimiz içindeki metinlerin kırmızı renkte görüntülenmelerini sağladık. Burada color özelliğinin yalnızca metinler üzerinde hakimiyet kuran bir özellik olduğunu hatırlatmak istiyorum.

2. Hexadecimal (Sayısal) Renk Tanımları

Oluşturduğumuz tasarımları hayatı geçirirken konu renge gelince elbette “renk tanımları” CSS ile uygulamada bizim için yetersiz kalacaktır.

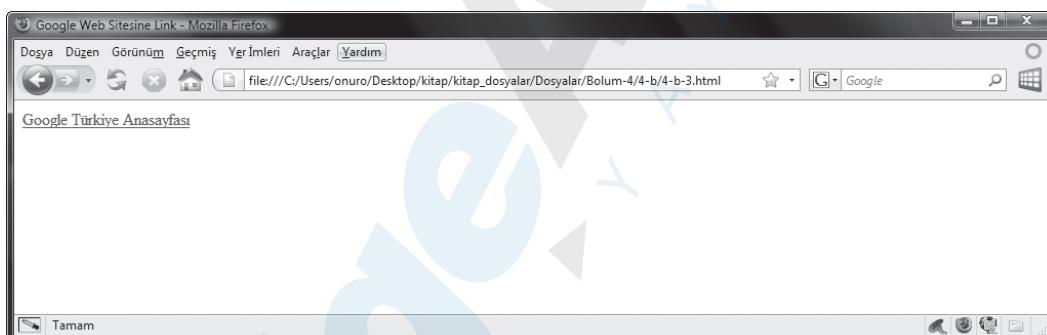
Web’de en popüler renk kullanım yöntemi olan Hexadecimal karakterlerden bahsetmek istiyorum. Hexadecimal karakterler, 6 haneli ve modüler aritmetikten gelen 16lı sayı sistemi baz alınarak kullanılan matematik tabanlı renk kodlarıdır.

Girilen 6 haneli renk kodunu, tarayıcı bilgisayarın ekran kartındaki eşdeğer monitör rgb tonuna çevirir ve bize o rengi sunar. Bu sayede çalışmamızda kullandığımız her rengi tasarım uygulama esnasında CSS içinde kullanabiliriz.

Hexadecimal renk kodları, renk tanımlarından farklı olarak başlarına diyez “#” yerleştirilerek değer olarak girilirler. Sıfır “0” rakamından başlayıp “F” harfinde sonlanırlar ve rakamlar ile harfler arası bu kombinasyon bize renkleri sunar.

Örnek:

```
a {  
    color: #AC0000;  
}
```



Figür 4.b.3

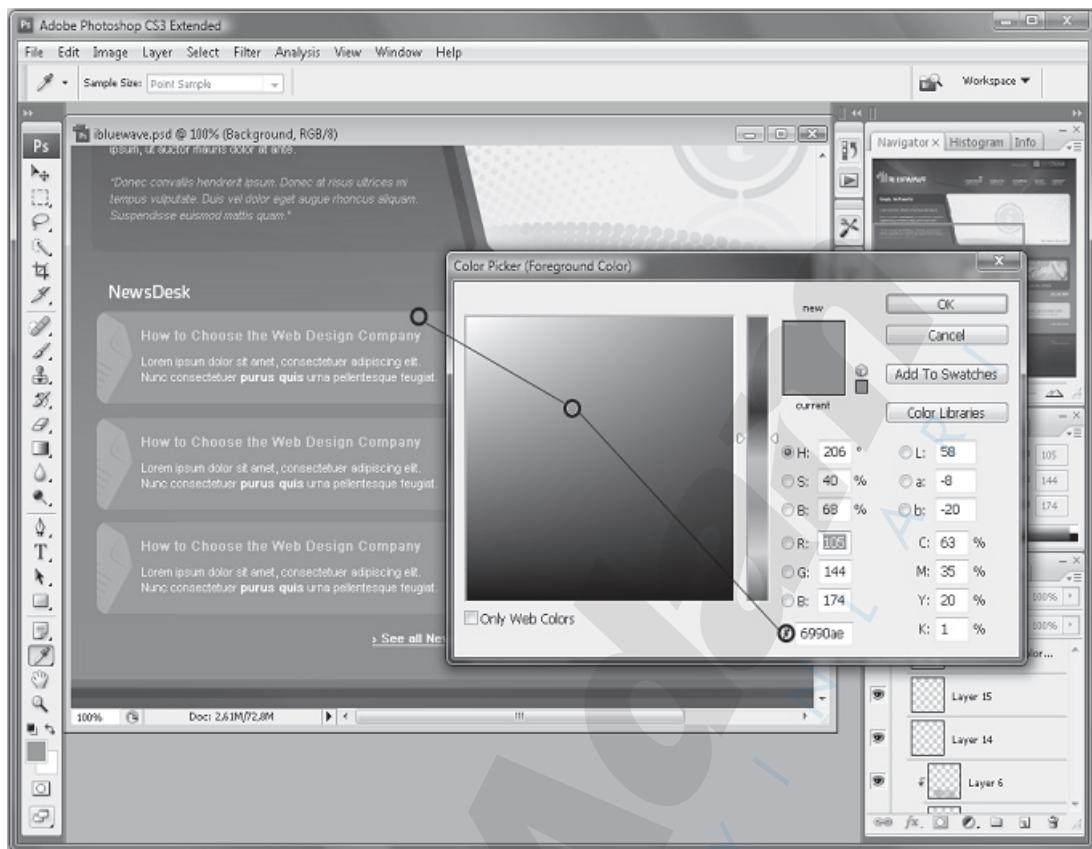
Bu renk kodu bize vişne kırmızısı rengini verecektir.

Kodların nasıl çalıştığını mantığını biraz açıklamaya çalışayım:

Hexadecimal renk kodlarında en koyu ton “0”a en açık “F”e denk gelir ve eğer renk kodu olarak 6 haneli olarak “000000” girecek olursanız **rgb skalasında** siyah, “FFFFFF” girecek olursanız da beyaz rengi elde edersiniz. 6 hanenin ilk iki hanesi rgb tonlarındaki kırmızıyı (r-red), orta iki hane yeşili (g-green), son iki hane ise maviyi (b-blue) temsil eder.

Elbette renk karşılıkları için renklerin hexadecimal kodlarını ezberlemek mümkün değildir. Ancak bu bizim için sorun olmayacağından emin olabiliriz. Zira gerek Adobe Photoshop’ta, gerekse Dreamweaver’da seçtiğimiz renk tonlarına karşılık gelen hexadecimal kodları görüntüleyerek kullanabilmemiz mümkün olacaktır.

Örneğin Photoshop ekranında hazırladığınız bir çalışmadan renk almaya çalıştığınıza farz edelim. Rengi **damlalıklıkla** seçtiğinden sonra renk paletini açacak olursanız, **renk skalasının hemen yanında alta** hexadecimal renk karşılığını görebilir, o kodu kopyalayarak CSS içindeki selektörünüzü renk olarak atayabilirsiniz:



Burada bir ipucu vermek istiyorum. Eğer 6 haneli renk kodunun **son 3** hanesi aynı değerleri içeriyorsa, bu son 3'ünü yazmadan da aynı etkiyi elde edebiliyoruz.

Örnek:

```
a {
  color: #000; /* #000000 kodunun kısa hali */
}
```

3. RGB Değerleriyle Renk Uygulama

Hexadecimal ve renk adı değerleri haricinde kullanabileceğimiz rgb renk değerleri de bulunmaktadır.

Bu yöntemde 0dan 255'e kadar olan red (kırmızı), green (yeşil) ve blue (mavi) tonları sırayla rakkamsal olarak girilerek ton elde edilmeye çalışılır.

Örnek:

```
a {
  color: rgb(0,0,0); /* siyah */
}
h1 {
  color: rgb(255,255,255); /* beyaz */
}
```

Burada 255'e kadar olmak üzere rakam girmek yerine rgb tonlarına yüzde (%) cinsinden de değer girebiliriz:

```
h1 {
  color: rgb(%100,%100,%100); /* beyaz */
}
```

A Etiketi İçin Linklere Özel Class'lar

Aslında bu konu renklerden çok selektör türleriyle alakalı olsa da, özellikle link renklendirme esnasında daha yoğun değineceğimiz için burada yer vermek istedim.

Link oluşturmamızı sağlayan [etiketi](#) (ve teoride diğer tüm block level elementler) 5 durum içerir. Bunlar linklere özel temel seviyede pseudo class'lar olarak adlandırılırlar. Bu class'lar ziyaretçinin elemente olan mouse veya klavye hareketine tepki verecek şekilde oluşturulmuştur. Bu da bize durum değişiminde farklı etkileri verebileceğimiz anlamına gelir.

Normal şartlarda durumlara özel pseudo class selektörünü açmamışsa 5 durum için de aynı özelliklerini tanımlıyoruz demektir. Pseudo class tanımlamaları selektör adından hemen sonra iki nokta üst üste ":" eklerek oluşturulurlar.

Şimdi bu 5 durumu içerek bir selektör oluşturup ne ifade ettiklerini belirtmeye çalışayım:

```
a {
  color: red;
}
```

Henüz 5 durumdan herhangi birini tanımlamış değilim. A etiketine uyguladığımız buradaki stillendirme, a etiketinin aşağıdaki tüm durumlarının da aynı olacağını belirtir. Onları ayrı ayrı değiştirmek istersek:

- 1.

```
a:link {
  color: red;
}
```

Herhangi bir mouse olayı gerçekleşmemişse linkin alacağı normal haldir. Yani mouse ile link üzerine henüz herhangi bir olay gerçekleştirmediysek linkimiz **kırmızı** renkte görünecektir.

- 2.

```
a:visited {
  color: blue;
}
```

Mouse ile bu linke daha evvel tıkladıysak, linkimiz artık normal görüntüleme esnasında tıklanmış olan diğerlerinden farklı olarak **mavi** renkte görünecektir.

- 3.

```
a:hover {
  color: gray;
}
```

Mouse'umuz ile linkin üzerine gelecek olursak linkin alacağı durumdur. Link metnimiz **gri** renkte görüntülenecektir.

```
4.
a:active {
color: purple;
}
```

Mouse'umuz ile linkin üzerine tıkladığımız anda linkin alacağı durumdur. Link metnimiz **mor** renkte görüntülenecektir.

```
5.
a:focus {
color: gray;
}
```

Mouse yerine klavyeyle siteyi gezen bir kullanıcı, Tab tuşunu kullanarak içerik ve link arasında gezme yapıyorrsa, link üzerine Tab etkin olduğunda linkin alacağı durumdur. Link metnimiz gri renkte görüntülenecektir.

Pseudo class'lar aslında çok daha geniş element seçici niteliklerine sahip. Ancak biz örnek uygulamalarımızı mümkün olduğunda temel seviyede tutarak aynı etkiyi almaya çalışacağımız için, tüm pseudo class niteliklerini kullanmayacağız.

4.c Web Dokümanları İçindeki Elementleri CSS ile Renklendirmek

Renk değerleri konusunda biraz bilgi sahibi olduğumuza göre, artık onları dokümanlarımızdaki daha fazla elemente atamaya başlayabiliyoruz.

Öncelikle dilerseniz biçimleyeceğimiz elementlerin yer aldığı basit bir HTML dokümanı oluşturalım:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns= "http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html charset=utf-8"/>
<title>Lorem Ipsum Nedir?</title>
<style type="text/css">
/* az sonra buraya css kodlarını gireceğiz*/
</style>

</head>
<body>

<div id="anakutu">
<p id="ilk_paragraf">
Lorem Ipsum, masaüstü yayıncılık ve basın yayın sektöründe
kullanılan taklit yazıbloğu olarak tanımlanır. Lipsum,
oluşturulacak şablon ve taslaklarda içerik yerine geçerek yazı
bloğunu doldurmak için kullanılır.
</p>
```

```

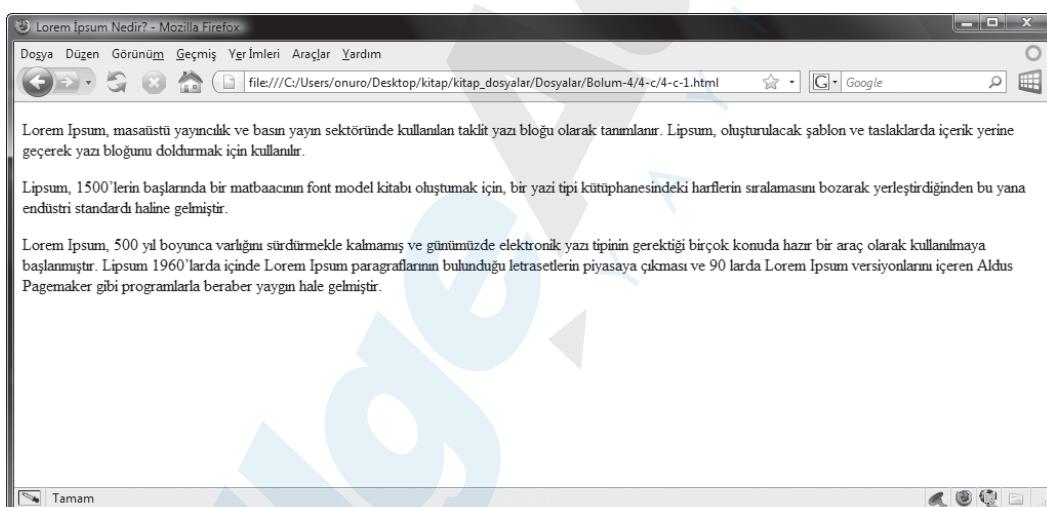
<p>
Lipsum, 1500'lerin başlarında bir matbaacının font model kitabı
oluştumak için, bir yazı tipi kütüphanesindeki harflerin
sıralamasını bozarak yerleştirdiğinden bu yana endüstri standarı
haline gelmiştir.
</p>

<p>
Lorem Ipsum, 500 yıl boyunca varlığını sürdürmekle kalmamış ve
günümüzde elektronik yazı tipinin gereği birçok konuda hazır
bir araç olarak kullanılmaya başlanmıştır. Lipsum 1960'larda
içinde Lorem Ipsum paragraflarının bulunduğu letasetlerin
piyasaya çıkması ve 90 larda Lorem Ipsum versiyonlarını içeren
Aldus Pagemaker gibi programlarla beraber yaygın hale gelmiştir.
</p>

</div> <!-- #anakutu kapanışı -->

</body>
</html>

```



Figür 4.c.1

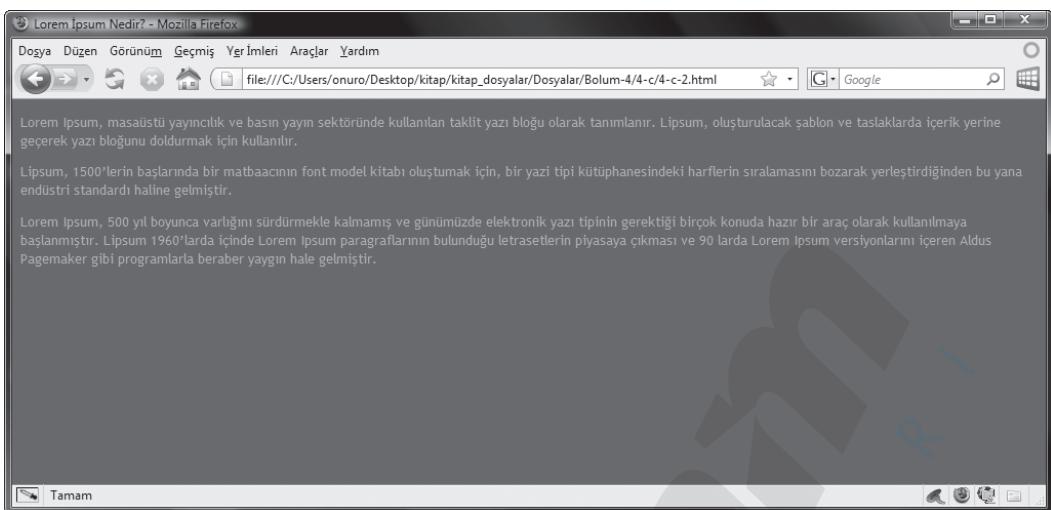
Anakutu adlı bir id selektörünü bir div elementine atadık ve o div içine 2 paragraf yerleştirdik. Paragraflardan ilkinin diğerlerine göre farklı olması için ona özel olarak “**ilk_paragraf**” id'si atadık.

Şimdi renklendirmeye yavaş yavaş geçelim. Öncelikle body elementini biçimleyerek doküman geneli renklerini verelim:

```

body {
background-color: #2e6673; /* koyu pastel mavi arkaplan */
color: #9bbab9; /* açık pastel mavi metin rengi */
font-family: "Trebuchet MS";
font-size: 9.5pt;
}

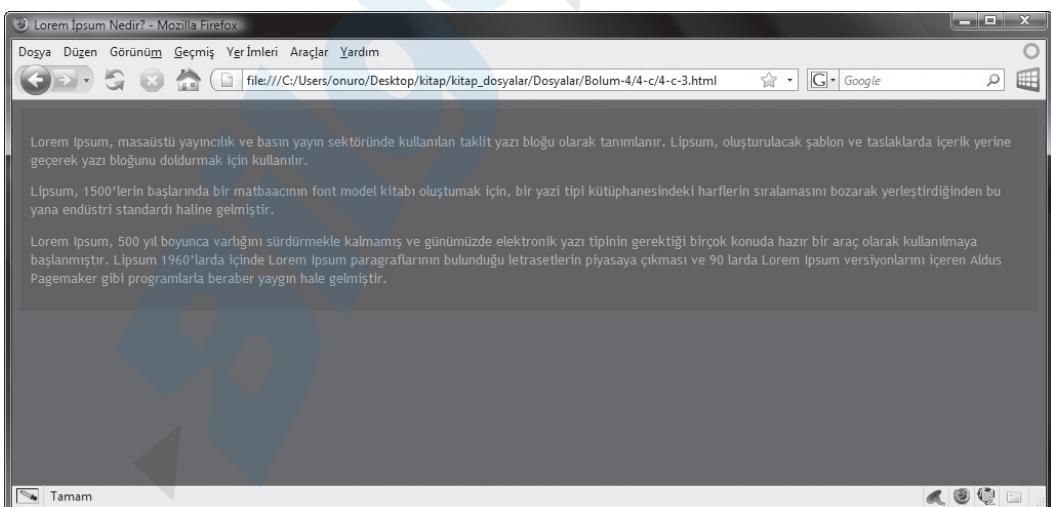
```

*Figür 4.c.2*

Tanımladığımız özelliklerle, doküman geneli arkaplan ve metin rengini belirlemiş olduk.

Şimdi de anakutu özelliklerini tanımlayalım:

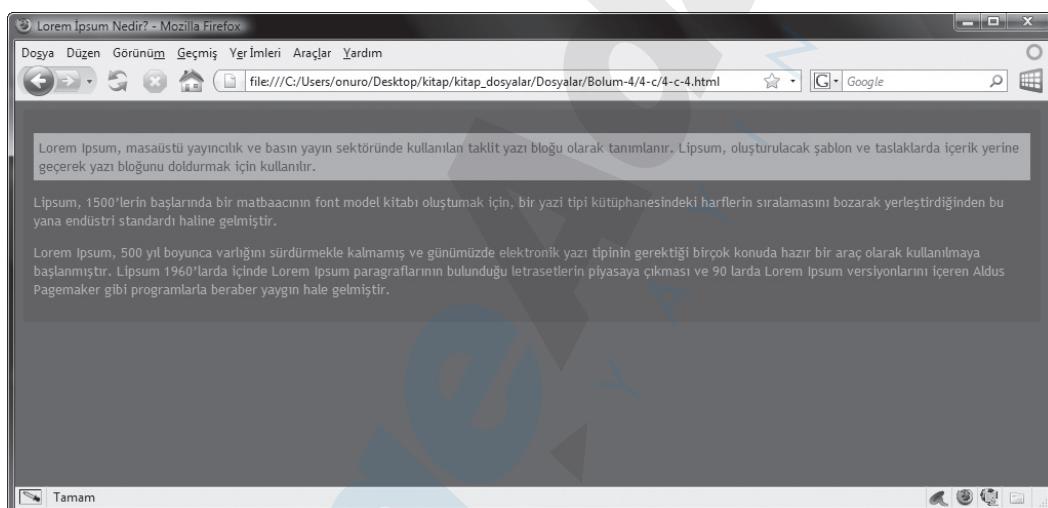
```
body {
background-color: #2e6673; /* koyu pastel mavi arkaplan */
color: #9bbab9; /* açık pastel mavi metin rengi */
font-family: "Trebuchet MS";
font-size:9.5pt;
}
#anakutu {
padding: 10px; background-color: #295c68;
}
```

*Figür 4.c.3*

Böylece kutu geneli olarak da doküman geneline göre daha koyu bir mavi arkaplan tonu atayarak ara kontrast oluşturduk ve okunaklığını biraz daha rahatlatmak için anakutuya iç boşluk oranı olarak 10 piksel padding oranı verdik.

Şimdi son olarak `ilk_paragraf` selektörümüzü biçimlendirerek ilk paragrafımızın görünümünü diğer paragraflardan ayırtıralım:

```
body {
background-color: #2e6673; /* koyu pastel mavi arkaplan */
color: #9bbab9; /* açık pastel mavi metin rengi */
font-family: "Trebuchet MS";
font-size: 9.5pt;
}
#anakutu {
padding: 10px; background-color: #295c68;
}
p#ilk_paragraf {
padding: 5px; background-color: #9bbab9; color: #295c68;
}
```



Figür 4.c.4

Fark ettiğiniz gibi ilk paragrafa atadığımız renkler, daha önceki renk kodlarıyla aynı. Ancak atadığımız noktalar farklı. Anakutu arkaplan rengi olan açık pastel maviyi paragraf metin rengi olarak belirlerken paragrafin arkaplan rengi olarak doküman geneli metin rengi olarak seçtik.

Özel stillendirme yaptığımız ilk paragraf haricindeki paragrafların özellikleri de elbette ebeveyn element olan body ve anakutu div'inden geliyor.

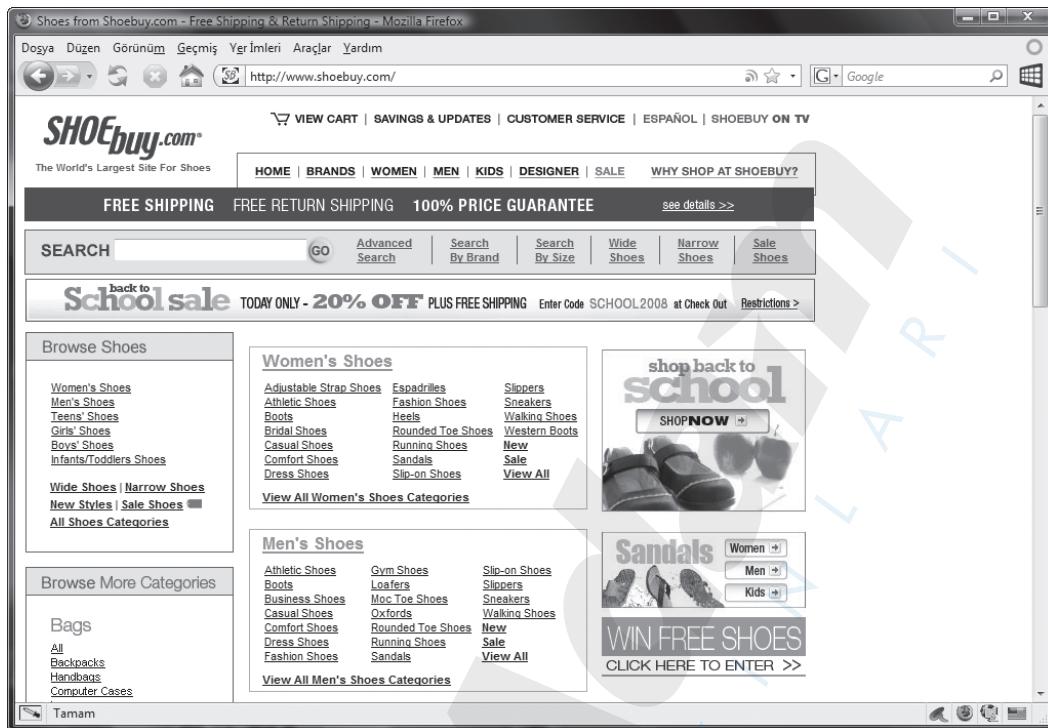
4.d CSS Renklendirme ile Derinlik Oluşturmak

Baba mesleği fotoğrafçılıktan gelen bir alışkanlık; yaptığım çalışmaların görsel kıstasları içinde derinlik önemli yer teşkil eder. Bu tüm tasarım sektörleri için geçerlidir diyebiliriz.

Nasıl ki boğaz manzarası konulu başarılı bir fotoğraf karesinde boğaz köprüsünün derinlik oranı arkasında kalan obje ve öğelere göre daha çarpıcı ve netse, Web tasarımda da göze odak noktası oluşturmak için renklerle derinlik oluştururuz (aslen bunun temel seviyede denemesini hafif kontrast kullanarak bir önceki örnekte yapmış olduk).

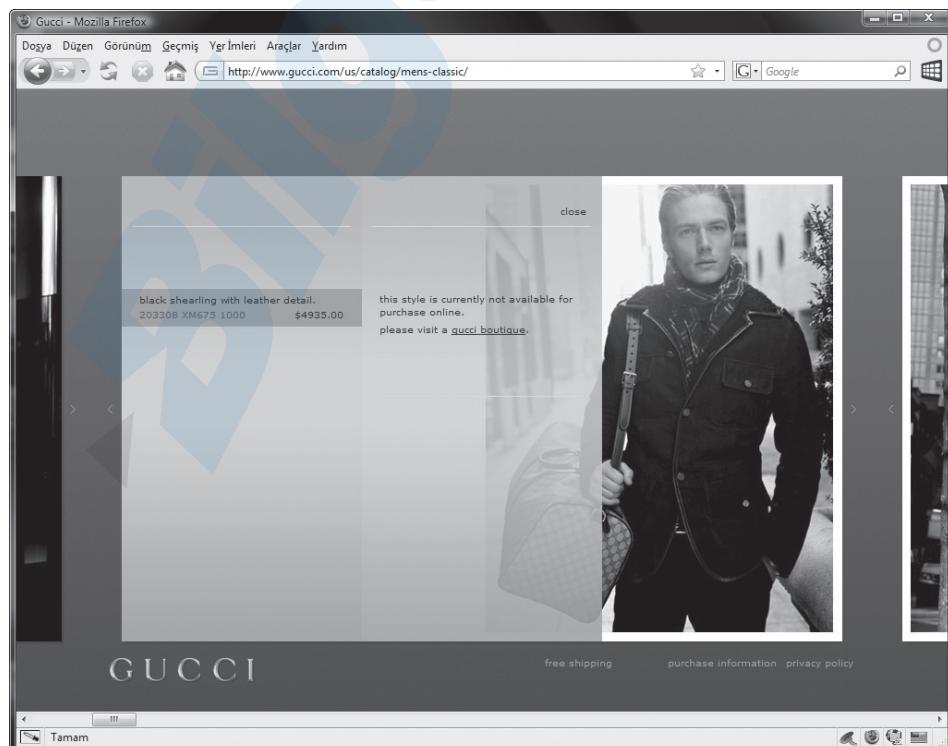
Derinliğin önemsenmediği durumlarda; özellikle alışveriş siteleri ve portallar gibi geniş içeriye sahip dokümanlar içeren sitelerin gezilmesi esnasında ziyaretçi tarafında sıkıntılar olabiliyor.

Buna örnek olarak renk ve görsel derinliğe pek sahip olmayan Shoebuy (www.shoebuy.com) sitesine bir göz atalım:



Öğelerin tutarsız dağılımı ve tonlama eksigi ziyaretçinin ilk başta sitede kaybolmasına neden olurken aradığı içeriğe ulaşması da biraz zaman alıyar olacak.

Ancak 2007 senesi Standartlara Uyumlu En İyi Tasarlanmış Web Sitesi olan Gucci (www.gucci.com/us/catalog/mens-classic/) alışveriş Web sitesi ise tam olarak az önce bahsettiğim kriterlerin başarılı uygulanmış haline sahip.



(Siteyi gezdiyorsanız; bazılarınıza Flash gibi görünecek olsa da aslen geçiş ve hareket efektleri Javascript uygulamalarından ibarettir. Bu site içerik, sunum ve davranışlar uygulamalar katmanının hepsine birden de örnek teşkil ediyor.)

Tonların uyumuna ve kontrast kullanılarak oluşturulmuş derinliğe (arkaplan, içerik renk dengesi) dikkat edelim.

Eminim sizin de bir gün hedeflerinizde bu kalitede çalışmalar üretmek yer alıyor.

Şimdi dilerseniz derinlik konusuna biraz da uygulama yaparak dejinelim.

Öncelikle bir önceki HTML dokümanımızın içeriğini örnek olarak baz alıp ikinci paragrafa da bir id selektörü atayarak üçüncü paragrafin ilk cümlesini yatkı görünecek hale getiriyorum:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns= "http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html charset=utf-8"/>
<title>Lorem Ipsum Nedir?</title>
<style type="text/css">
/* az sonra buraya css kodlarımızı gireceğiz*/
</style>

</head>
<body>

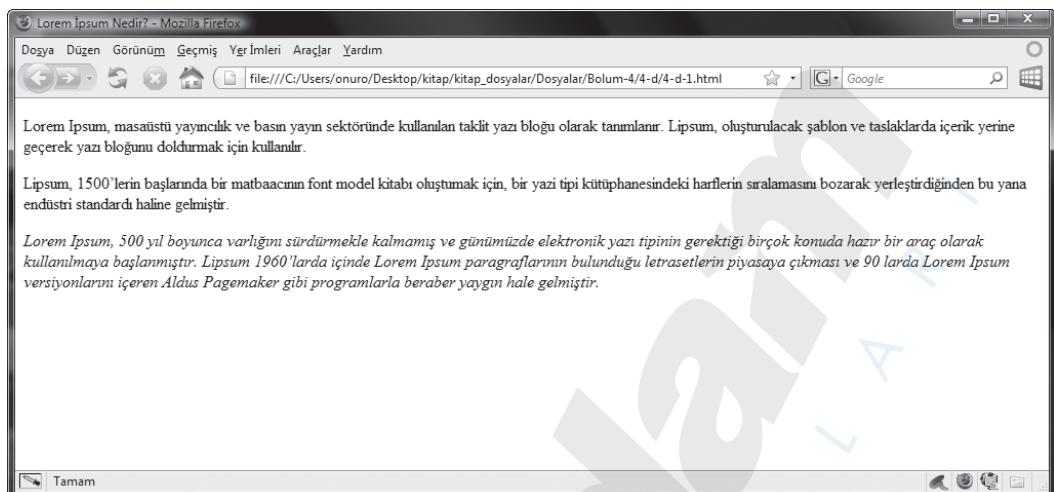
<div id="anakutu">
<h1> Lorem Ipsum Nedir?</h1>
<p id="ilk_paragraf">
Lorem Ipsum, masaüstü yayıncılık ve basın yayın sektöründe
kullanılan taklit yazıbloğu olarak tanımlanır. Lipsum,
oluşturulacak şablon ve taslaklarda içerik yerine geçerek yazı
bloğunu doldurmak için kullanılır.
</p>

<p id="ikinci_paragraf">
Lipsum, 1500'lerin başlarında bir matbaacının font model kitabı
oluştumak için, bir yazı tipi kütüphanesindeki harflerin
sıralamasını bozarak yerleştirdiğinden bu yana endüstri standarı
haline gelmiştir.
</p>

<p>
<em>Lorem Ipsum, 500 yıl boyunca varlığını sürdürmekle kalmamış ve
günümüzde elektronik yazı tipinin gerektiği birçok konuda hazır
bir araç olarak kullanılmaya başlanmıştır.</em> Lipsum 1960'larda
içinde Lorem Ipsum paragraflarının bulunduğu letasetlerin
piyasaya çıkması ve 90 larda Lorem Ipsum versiyonlarını içeren
Aldus Pagemaker gibi programlarla beraber yaygın hale gelmiştir.
</p>
```

```
</div> <!-- #anakutu kapanışı -->

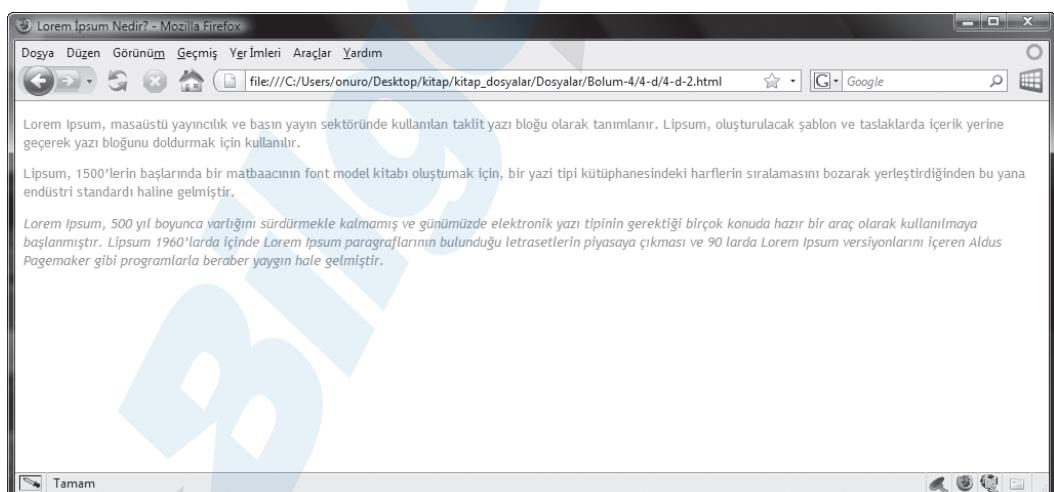
</body>
</html>
```



Figür 4.d.1

Dokümanı şimdilik derinliğe sahip olmayacak şekilde stillendirelim:

```
body {
color: #999; /* gri metin rengi */
font-family: "Trebuchet MS";
font-size: 9.5pt;
}
```



Figür 4.d.2

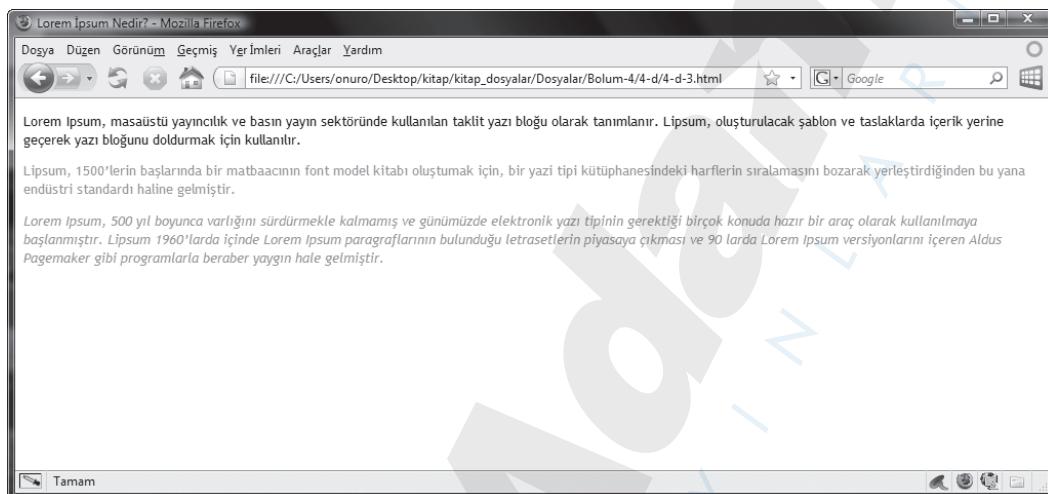
Doküman geneli özelliklerine atadığımız değerlerle tüm metinlerin gri tonda görüntülenmesini sağladık. Bu şu an için ne albeni, ne de görsel derinlik taşımıyor. Bunu paragraflarımıza özel tonlamalar yaparak değiştireceğiz.

İlk paragrafımızın metin rengi üzerinde bir değişiklik yapalım:

```

body {
color: #999; /* gri metin rengi */
font-family: "Trebuchet MS";
font-size:9.5pt;
}
#ilk_paragraf {
color:#000; /* siyah metin rengi */
}

```



Figür 4.d.3

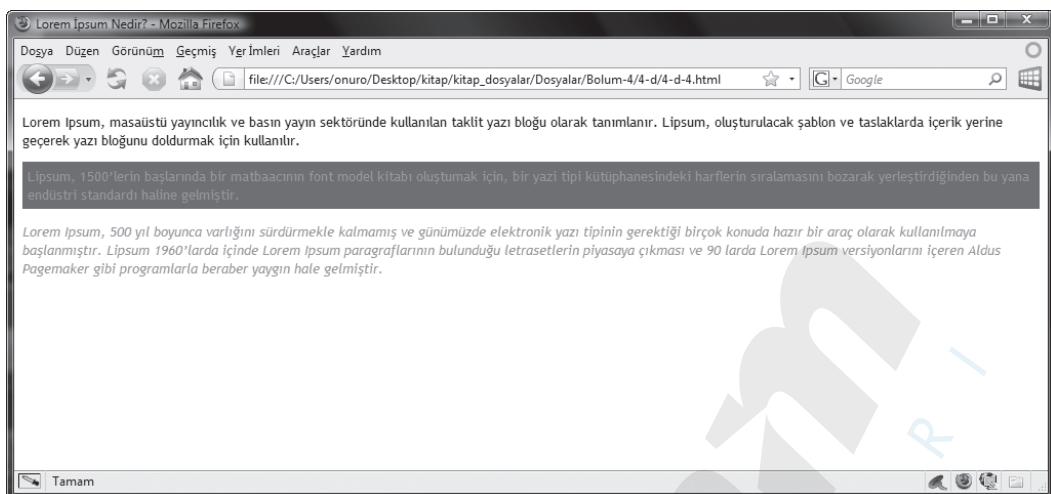
İlk paragraf diğer içerik bloklarından tamamen ayrıntı değil mi?

Bu ayrışmayı biraz daha netlestirelim:

```

body {
color: #999; /* gri metin rengi */
font-family: "Trebuchet MS";
font-size:9.5pt;
}
#ilk_paragraf {
color:#000; /* siyah metin rengi */
}
#ikinci_paragraf {
background-color:#666; /* koyu gri metin rengi */
padding:5px; /* 5 piksellik paragraf iç boşluğu */
}

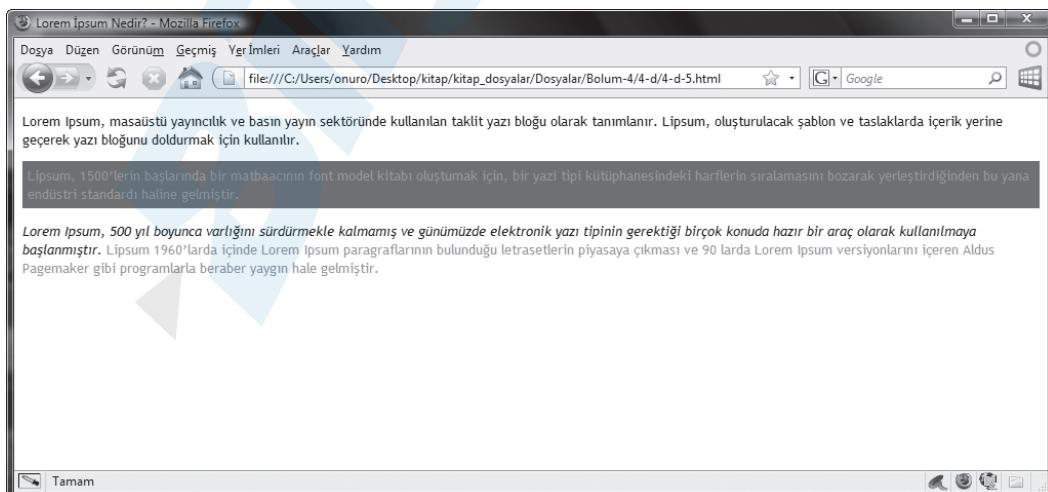
```



Figür 4.d.4

Ayrıntı biraz daha belirginleştirdik. Peki, derinliği biraz daha artıralım:

```
body {
    color: #999; /* gri metin rengi */
    font-family: "Trebuchet MS";
    font-size: 9.5pt;
}
#ilk_paragraf {
    color: #000; /* siyah metin rengi */
}
#ikinci_paragraf {
    background-color: #666; /* koyu gri metin rengi */
    padding: 5px; /* 5 piksellik paragraf iç boşluğu */
}
em {
    color: #000; /* siyah metin rengi */
}
```



Figür 4.d.5

Son paragrafın ilk satırı da ilk paragrafın görsel gücüne sahip oldu. Böylece ziyaretçinin gözü ilk olarak ilk ve son paragrafa odaklanacak, sonra da ikinci paragraf içeriğine yönelecek.

İçeriğin belli alanlarını diğer alanlardan ayırtırarak CSS yardımıyla tasarımlarımızda kolaylıkla derinlik oluşturabilir, ziyaretçimize öncelikle nereye odaklanması gerektiğini görsel bir dille söyleyebiliriz.

4.e Kontrast Yönetimi

Tasarladığımız Web sitelerinin ziyaretçilerin gözü yormadan kolay gezilebilir bir görsellikte olması gerektiğini babetmiştim. Şimdi, doğru renk ton aralıklarının seçilebilmesi ve uygun kontrasta renk yerleşimi için birkaç ipucu vermek istiyorum.

Worldwide Web Konsorsiyumu'nun (W3C – www.w3c.org) tanımladığı erişilebilir Web tasarım rehberi de bu konuda uyarılar belirtiyor.

Bunlardan en önemlisi, arkaplan grafik ve renkleri ile içerik metinlerinin mümkün olduğunda ayırtması gereki̇ği gezilebilirliği kolaylaştırması gereki̇ği yönünde. Hatta bu konuda bir ışık kontrast oranı oluşturmaya yönelik bir tabloları da mevcut (Luminosity Contrast Ratio). Aşağıdaki adresten bu rehbere ve tabloya ulaşabilirsiniz:

<http://www.w3c.org/TR/WCAG20/guidelines.html#visual-audio-contrast>

Doğu kontrasti elde edebilmek için daha farklı araçlar da mevcuttur. Bunlardan bir tanesi de Juicy Studio'nun Luminosity Contrast Ratio aracıdır. Aşağıdaki adresten ulaşarak içerik ve arkaplan renklerini girerek kontrast oranının uyumunu test edebilirsiniz:

<http://www.juicystudio.com/services/luminositycontrastratio.php>

Buraya kadarki aşamalardan tasarım adına anladıklarımızı şöyle bir listeleyelim:

- Hazırladığımız tasarımların öncelikle içeriğini HTML olarak oluşturuyoruz, sonra CSS ile hâyata geçiriyoruz.
- Tasarımı oluştururken tipografik önceliklere ve yazıtının genel tasarımla bütünlüğüne dikkat ediyoruz.
- Derinlikler ve odak noktaları oluşturarak ziyaretçimizin gezdiği siteyi daha kolay ve erişilebilir bir şekilde görüntülemesine yönelik ince ayarlar yapıyoruz.

Bundan sonraki aşamada biraz daha ileri gideceğiz ve tasarımlarımızı hayatı geçirirken birebir etkiyi elde etmek için grafikler ve kutusal yapıyı nasıl dokümanlarımıza kombine ederiz, bunların çözümüne degeineceğiz.



5 CSS ile Tasarımda Renk Uygulama

5 CSS ile Tasarımda Renk Uygulama

- Bir HTML Elementine Arkaplan Grafiği Atamak
- Arkaplan Repeat (Tekrar) Özellikleri
- Arkaplan Pozisyonlandırma
- Fixed (Sabit) veya Scroll (Kaydırılabilen) Arkaplan Özellikleri

CSS ile Tasarımda Renk Uygulama

5.a Bir HTML Elementine Arkaplan Grafiği Atamak

Şu ana kadar, arkaplan tonları üzerinde işlem yaptık ancak grafikleri ile neler yapabileceğimize dair uygulamalara henüz girişmemiştik. Bu bölümde CSS ile arkaplan grafikleri üzerinde duracağız.

Öğelerimize arkaplan grafikleri uygulayarak onları görsel olarak daha albenili hale getirip etkili içerik görselleştirmeleri yapacağız.

Bildiğiniz gibi, öğelerin arkaplan özelliklerini biçimlemek için “background” özellik grubunu kullanıyoruz. Background özellik grubu, bize bir ögenin arkaplan rengini verebilmekle beraber, o ögeye arkaplan grafiği atamak, tekrar oranını belirlemek ve atanın arkaplan grafiğinin pozisyonunu belirlemek gibi harika ek biçimleme olanakları sunuyor.

Burada bir noktaya değinmek istiyorum:

Arkaplan grafikleri, CSS tarafında selektörlerle, dolayısıyla HTML içindeki elementlere atayarak oluşturduğumuz için, o grafiklere tıklamak, dolayısıyla HTML içindeki etiketindeki gibi işlemeler yaptırmamız mümkün olmayacak.

Dolayısıyla hangi grafiklerin arkaplan grafiği, hangilerinin içerik grafiği olduğunu konusunda zamanı geldiğinde doğru tercihleri yapıyor olmanı gerekecek.

Keza örneğin bir kitap yazarının biyografisinin bulunduğu bir div içerisinde yazarın portre fotoğrafı içerik grafiği olarak etiketiyle verilmesi gerekirken, fotoğrafın ve biyografi metninin bulunduğu bu div'in arkaplan grafiğini ççekler olarak farz edersek, bu ççeklerin arkaplan grafiği olarak CSS ile atanmaları gerekecektir.

Bir HTML Elementine Arkaplan Grafiği Uygulama

CSS arkaplanlar ile ilgili genel bilgiyi edindiğimize göre, dilerseniz şimdilik bir örnek olarak doküman geneli arkaplan grafiği oluşturalım ve onu daha evvelki html dokümanımızın arkaplanı olarak CSS ile yerlestirelim:

```
<DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  
<html xmlns= "http://www.w3.org/1999/xhtml">  
  <head>  
    <meta http-equiv="Content-Type" content="text/html charset=utf-8"/>  
    <title>Lorem Ipsum Nedir?</title>  
    <style type="text/css">  
      /* az sonra buraya css kodlarınıza gireceğiz */  
    </style>  
  
  </head>  
  <body>  
  
    <div id="anakutu">  
      <h1> Lorem Ipsum Nedir?</h1>
```

```

<p id="ilk_paragraf">


Lorem Ipsum, masaüstü yayıncılık ve basın yayın sektöründe kullanılan taklit yazıbloğu olarak tanımlanır. Ipsum, oluşturulacak şablon ve taslaklarda içerik yerine geçerek yazıbloğunu doldurmak için kullanılır.


</p>

<p id="ikinci_paragraf">

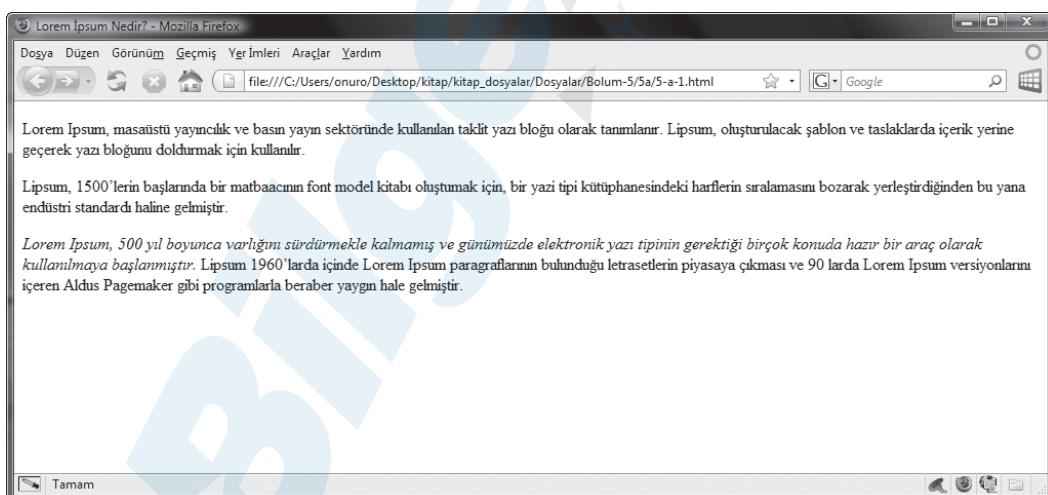

Lipsum, 1500'lerin başlarında bir matbaacının font model kitabı oluşturmak için, bir yazı tipi kütüphanesindeki harflerin sıralamasını bozarak yerleştirdiğinden bu yana endüstri standartı haline gelmiştir.


</p>

<p>
<em>Lorem Ipsum, 500 yıl boyunca varlığını sürdürmekle kalmamış ve günümüzde elektronik yazı tipinin gerektiği birçok konuda hazır bir araç olarak kullanılmaya başlanmıştır.</em> Lipsum 1960'larda içinde Lorem Ipsum paragraflarının bulunduğu letasetlerin piyasaya çıkması ve 90 larda Lorem Ipsum versiyonlarını içeren Aldus Pagemaker gibi programlarla beraber yaygın hale gelmiştir.
</p>

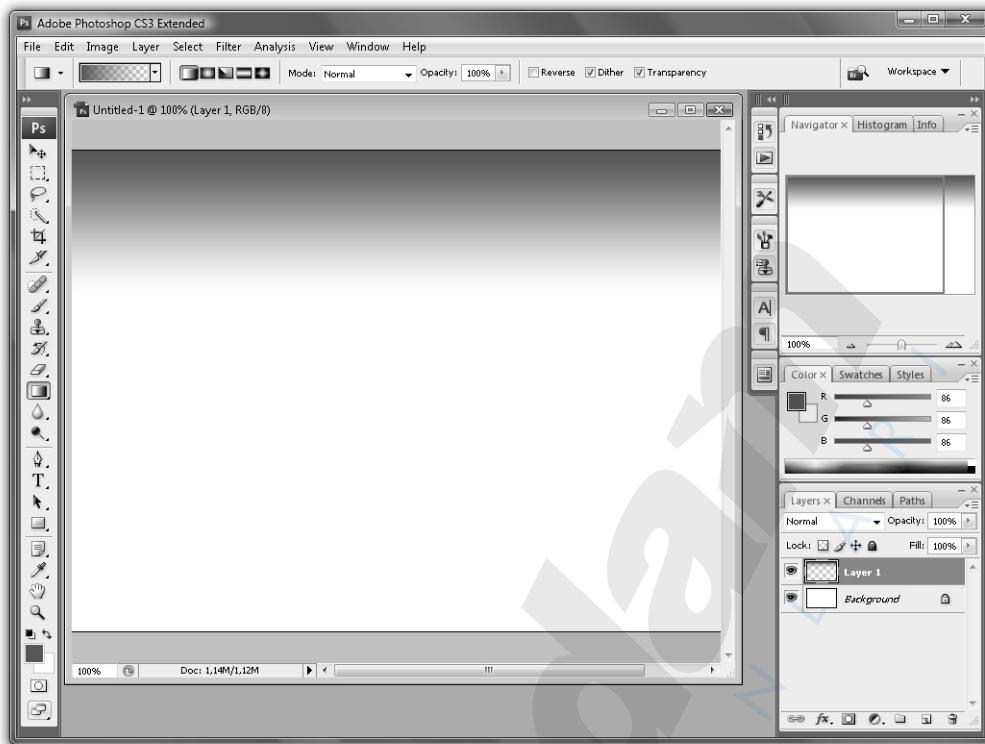
</div> <!-- #anakutu kapanışı -->
</body>
</html>

```



Figür 5.a.1

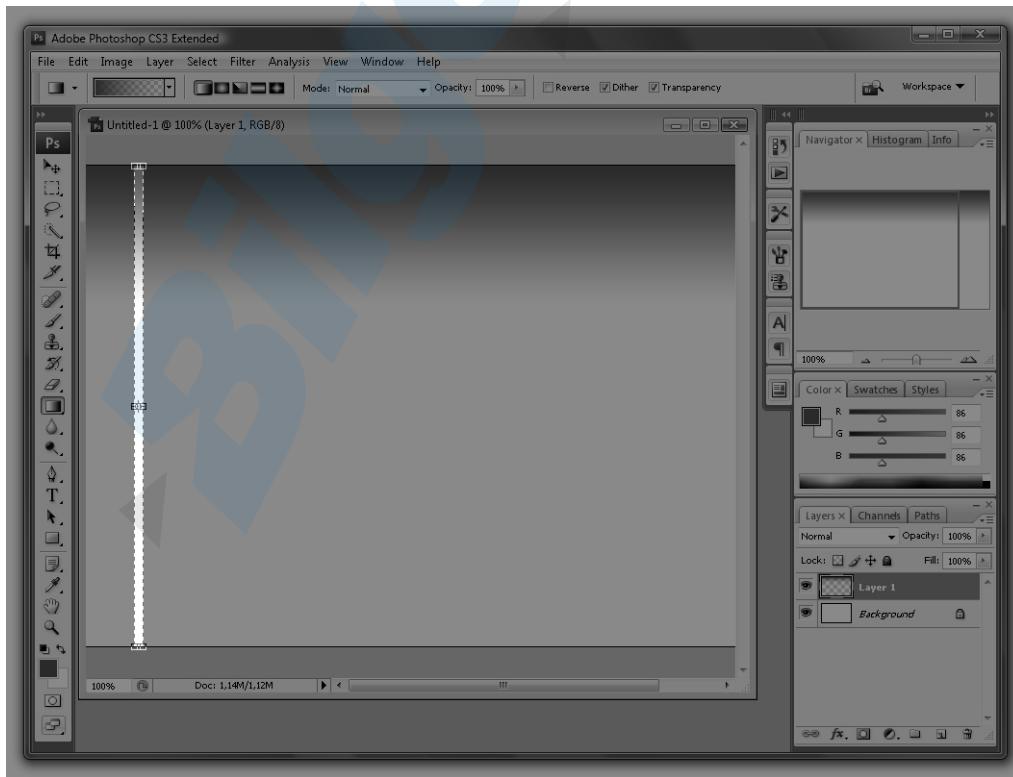
İlk iş olarak bu dokümanın arkaplanı için Photoshop ile griden beyaza yatay gidecek şekilde degradé olusturalım. Benim açtığım dosya 800 px genişliğinde ve 500 px yüksekliğe sahiptir:



Figür 5.a.2

800 px uzunlukta bir alana bu işlemi uygulamamın sebebi, bu grafik doküman arkaplanı olarak atandığında nasıl görünüğe dair fikir sahibi olmak. Yoksa elbette ki bu ebatlarda bir grafik dokümanınız için devasa bir büyülüğe ve uzun bir yükleme sürecine sahip olacaktır. İstediğimiz bu değil. Amacımız, bu geçişken degradenin kısa bir parçasını kesit olarak alıp, CSS ile yatay olarak tekrar ettirmek.

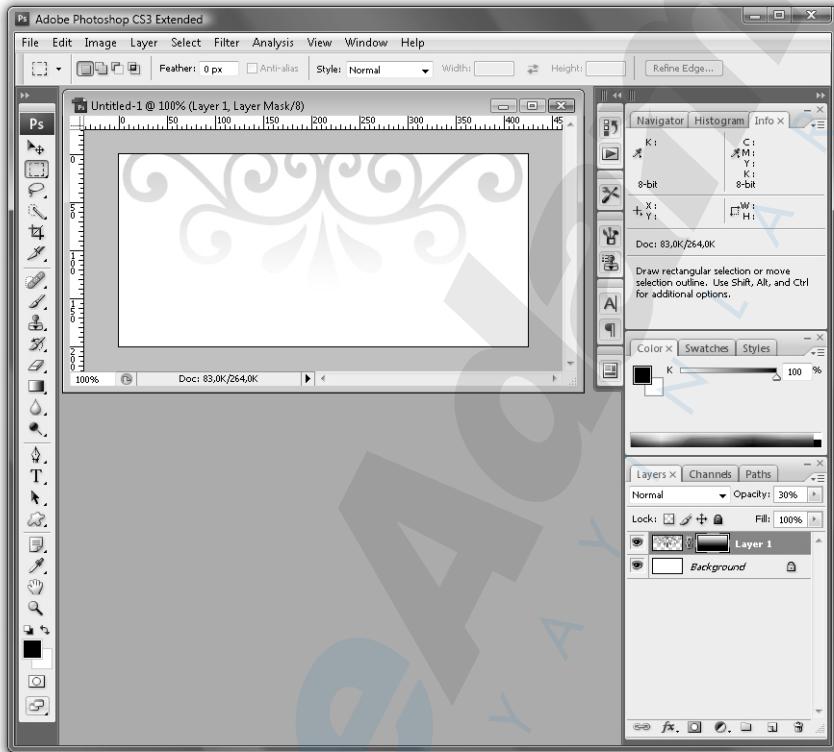
Bu nedenle 10 px'lik bir alanı croplayarak grafiği “arkaplan.jpg” adıyla html dokümanımızı oluşturduğumuz klasöre kaydediyoruz.



Figür 5.a.3

Şimdi sıra bu kısa grafiğin tekrar ederek HTML dokümanında arkaplan olarak yerleştirilmesine geldi. Doküman geneli için açacağımız selektör her zamanki gibi body etiket selektörü:

```
body {
    font: 9.5pt "Trebuchet MS";
    background:url(arkaplan.jpg) repeat-x #fff;
}
```



Figür 5.a.4

Yaptığımız bu işlemle sırasıyla şunları uygulamış olduk:

- Font özellik grubu kullanarak yazıtipini **9.5** punto ebadıyla “**Trebuchet MS**” olarak belirle.
- Body etiketi için arkaplan grafiği olarak arkaplan.jpg dosyasını kullan (url yöntemi)
- Bu grafiği **yatay olarak** doküman boyunca sınırsız olarak tekrar et (repeat-x). Dikey olarak tekrar ettirmek isteseydik (ki amacımız bu değil) repeat-y kullanmamız gerekecekti.
- Grafik yalnızca yatay olarak tekrar edeceğini doküman eğer 500 px'den sonraki uzunluğa erişirse (icerığın oranıyla doğru orantılı olarak doküman scroll'u uzayabilir) dikey uzunluğu sona erdiği andan itibaren beyaz ton arkaplan rengi kullan (#fff).

Kullandığımız özellik grubu, background. Dolayısıyla kısayol kullanmış olduk. Her opsiyonu ayrı ayrı background-image, background-repeat ve background-color özelliklerine atamalar yaparak uzun yöntemle de uygulayabilirdik.

5.b Arkaplan Repeat (Tekrar) Özellikleri

Aslında bir önceki örneğimizde bu özelliğin uygulanışına kısaca göz atmıştık. Şimdi arkaplan tekrar özellikleri konusuna biraz daha detaylı değinmek istiyorum.

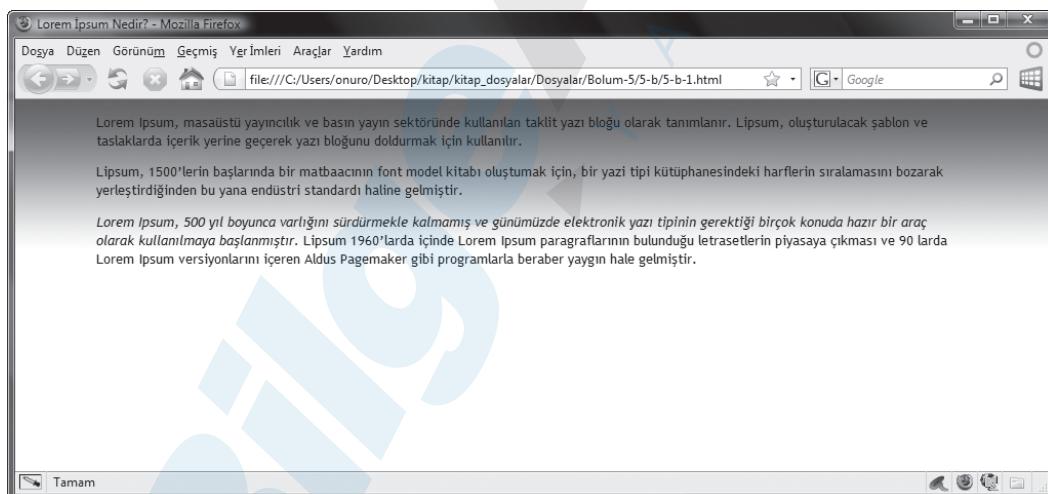
CSS özellikleri, bize bir arkaplanın yatay veya dikey olarak tekrarını sağlayabilir, veya tekrarlanmadan tek öğe olarak yerleştirilmesine izin verebilir. `background-repeat` özelliğine gireceğimiz değerlerle hangisini seçtiğimizi tanımlayabilir veya bir önceki uygulamadaki gibi `background` özellik grubu içinde bunu tanımlayabiliriz.

`background-repeat` özelliği 3 değer alabilir:

- **repeat-x**: yatay olarak tekrar et
- **repeat-y**: dikey olarak tekrar et
- **no-repeat**: tekrar gösterme, tek öğe olarak yerleştir.

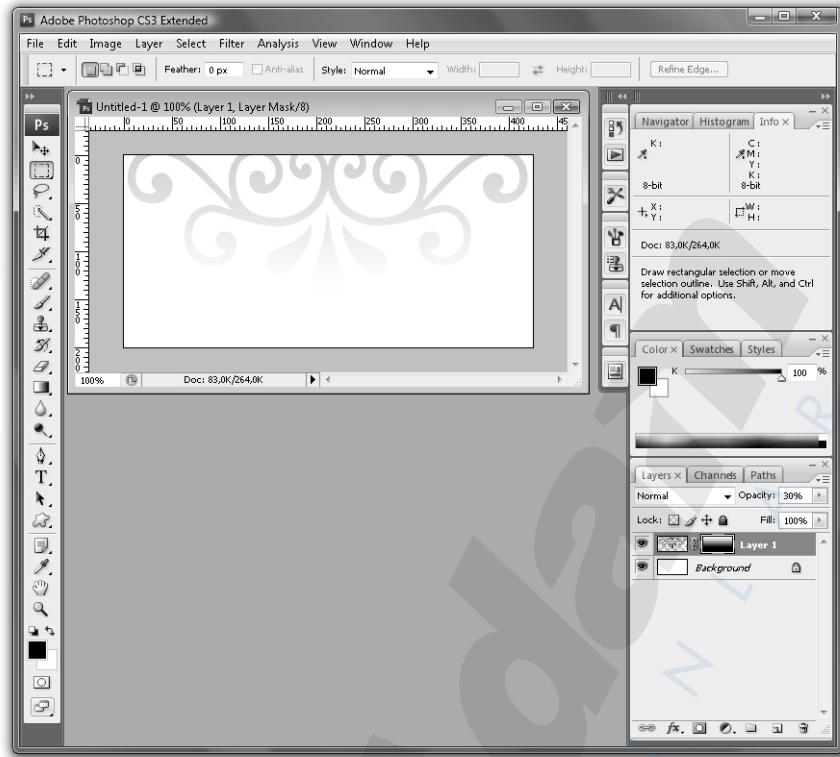
Şimdi dilerseniz bir önceki çalışmamızdan devam edelim ve anakutu div'ini 850px genişliğine sınırlayarak doküman geneline ortalayalım, zira az sonra Photoshop ile onun tepesinde yerleşik olarak duracak bir grafik oluşturup anakutu üzerine tekrar etmeyecek şekilde arkaplan grafiği olarak atayacağımız:

```
body {
    font: 9.5pt "Trebuchet MS";
    background:url(arkaplan.jpg) repeat-x #fff;
}
#anakutu {
    width: 850px;
    margin: 0 auto;
}
```



Figür 5.b.1

Şimdi Photoshop ile 425px genişliğinde ve 250px yüksekliğinde yeni bir doküman oluşturarak floral bir grafik yerleştirip "anakutu_bg.jpg" adıyla kaydediyoruz:



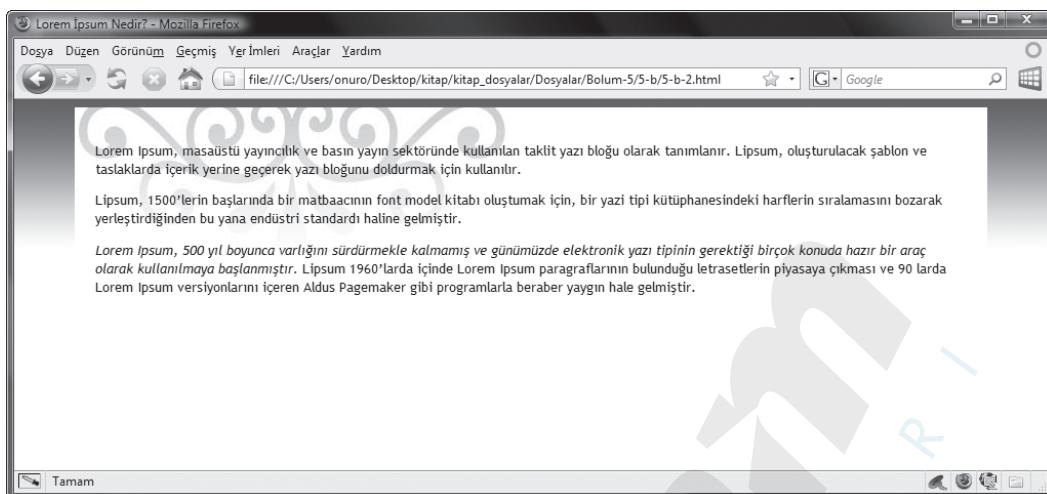
Peki, neden anakutu div genişliğimiz 850 piksel iken grafiğimiz **425** piksel, yani yarı genişliğine sahip?

Sebebi çok basit. Birincisi, 850 px genişliğinde uzunca tek grafik yükleme boyutu olarak biraz büyük. İkincisiyse bu büyülü engelini bir sonraki bölümde yarı genişliğindeki grafiği anakutu içine ortalayarak çözeceğiz.

Bu kez uygulayacağım örnekte ben anakutu için arkaplan özelliklerini kısayol kullanmadan ayrı ayrı gireceğim:

```

body {
  font: 9.5pt "Trebuchet MS";
  background:url(arkaplan.jpg) repeat-x #fff;
}
#anakutu {
  width: 850px;
  margin: 0 auto; /* ögeyi dokumana yatay olarak ortala */
  padding: 20px; /* 20 piksellik iç boşluk oranı */
  background-image:url(anakutu_bg.jpg); /* arkaplan grafiği
  yerleştir */
  background-repeat: no-repeat; /* grafiği tekrar etme */
  background-color: #fff; /* arkaplan rengi olarak beyaz renk
  göster */
}
  
```



Figür 5.b.2

Böylece son durum itibariyle, hem dokümanımıza, hem de anakutu div'imize birer arkaplan yüklemesi yaptık.

Bana sorarsanız anakutuda varsayılan olarak sol üste tekrar etmeyecek şekilde yerleşmiş olan grafiğimiz tam olarak mükemmel konumda durmuyor. Bir sonraki bölümde onu anakutu içinde ortalamayı göreceğiz.

5.c Arkaplan Pozisyonlandırma

CSS bize arkaplan grafiğimizin yer aldığı elementin hangi noktasına konumlandırmak istediğimizi dair biçimlemelere olanak tanıyor.

Dolayısıyla bir elemente arkaplan olarak atadığımız grafiği element içinde dikey ve yatay konumlarda background-position özelliğine girilecek değerler olarak aşağıdaki şekillerde uygulayabiliyoruz:

- **top:** üstte tut
- **left:** solda tut
- **bottom:** alta tut
- **right:** sağda tut

Bunların haricinde sırasıyla top ve left değerlerine **pozitif** ve **negatif** piksel (px) değeri girerek de arkaplanımızı konumlandıabilirmiz.

Normal şartlarda herhangi bir arkaplan konum özelliği girmezsek, tarayıcı standardında grafiğimiz sol üst (top left) konuma yerleşecektir.

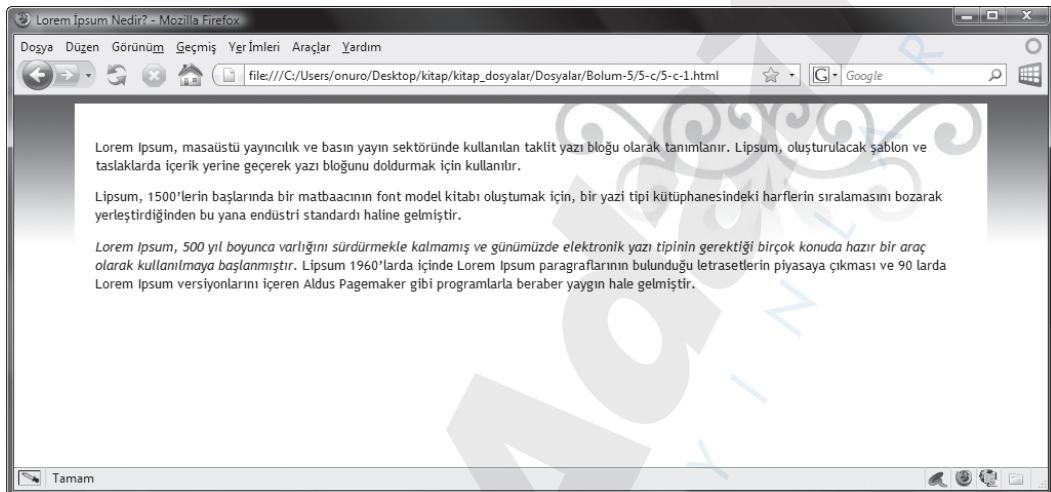
Şimdi bir örnek uygulamayı anakutu div'imizeındaki arkaplan grafiğini biraz daha mesafeli konumlandırarak yapalım:

```
body {
    font: 9.5pt "Trebuchet MS";
    background:url(arkaplan.jpg) repeat-x #fff;
}
#anakutu {
    width: 850px;
```

```

margin: 0 auto; /* ögeyi dokumana yatay olarak ortala */
padding: 20px; /* 20 piksellik iç boşluk oranı */
background-image:url(anakutu_bg.jpg) ; /* arkaplan grafiği
yerleştir */
background-repeat: no-repeat; /* grafiği tekrar etme */
background-color: #fff; /* arkaplan rengi olarak beyaz renk
göster */
background-position: top right; /*üst sağa yerleştir*/
}

```



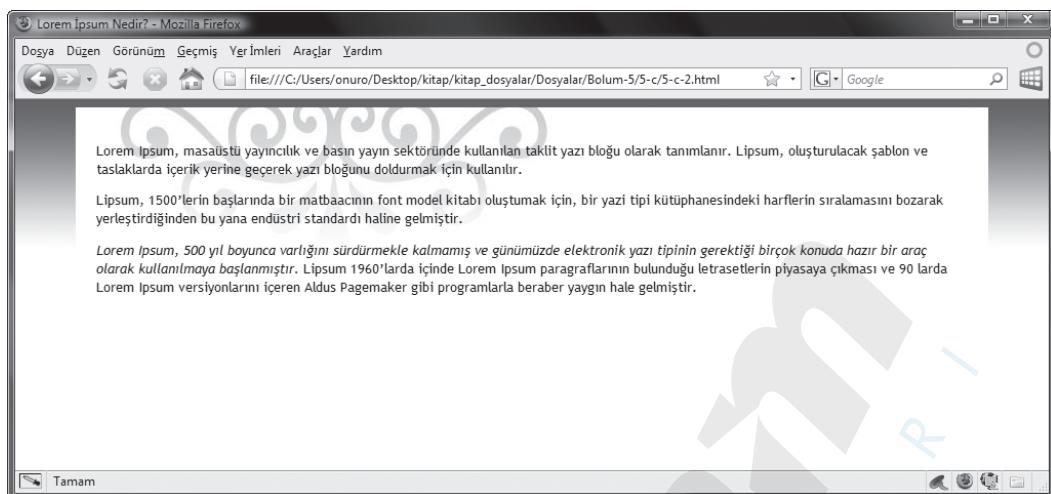
Figür 5.c.1

Görüldüğü gibi anakutu içindeki arkaplan grafiğimiz üst sağ tarafa yerleşti. Sağa konumlandırmak yerine, yine üstte tutup bu kez soldan 40 piksel mesafede tutmak istersek:

```

#anakutu {
width: 850px;
margin: 0 auto; /* ögeyi dokumana yatay olarak ortala */
padding: 20px; /* 20 piksellik iç boşluk oranı */
background-image:url(anakutu_bg.jpg) ; /* arkaplan grafiği
yerleştir */
background-repeat: no-repeat; /* grafiği tekrar etme */
background-color: #fff; /* arkaplan rengi olarak beyaz renk
göster */
background-position: 40px top; /*üste ve 40px mesafeye
konumlandır*/
}

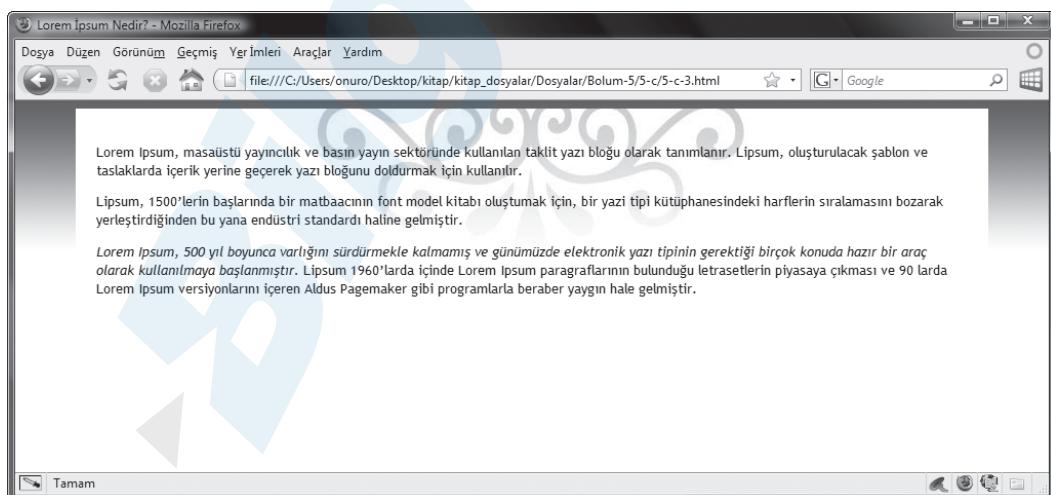
```



Figür 5.c.2

Dileseydik top konumu yerine de piksel değeri girerek yukarıdan da mesafe verebilirdik. Şimdi asıl yapmak istediğimiz konumlandırmayı yani grafiği üst orta konuma uygulayalım:

```
#anakutu {
    width: 850px;
    margin: 0 auto; /* ögeyi dokümana yatay olarak ortala */
    padding: 20px; /* 20 piksellik iç boşluk oranı */
    background-image:url(anakutu_bg.jpg); /* arkaplan grafiği
yerleştir */
    background-repeat: no-repeat; /* grafiği tekrar etme */
    background-color: #fff; /* arkaplan rengi olarak beyaz renk
göster */
    background-position: top center; /*üst ortaya konumlandır*/
}
```



Figür 5.c.3

CSS arkaplanlar ile elementlere bu şekilde arkaplan grafikleri atayabiliyor, onları elementler içinde dilediğimiz noktalara konumlandıtabiliyoruz.

5.d – Fixed (Sabit) veya Scroll (Kaydırılabilen) Arkaplan Özellikleri

Bazen içeriği bol ve scroll eden (dikey olarak uzayan) bir dokümanda, tek bir arkaplan grafiği kullanırken o grafiğin sabit bir konumda kalmasını, belli bir scroll mesafesinden sonra kaybolmasını önleyebilir veya doküman gibi onun da scroll etmesini sağlayabiliyoruz.

Bu işlemi background-attachment özelliğine atayacağımız aşağıdaki iki değer türüyle uygulayabiliyoruz:

- **fixed**: sabit yerleşim.
- **scroll**: scroll edebilen yerleşim

Örneğin bizim bir önceki çalışmamızdaki doküman geneli arkaplanını sabit (fixed) konumda tutmaya çalışalım. Bu işlem için doküman içeriğini kopyalayarak artırıp scroll etmesini sağlıyorum ve background-attachment özelliğine fixed değerini uyguluyorum:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns= "http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html charset=utf-8"/>
<title>Lorem Ipsum Nedir?</title>
<style type="text/css">
body {
font: 9.5pt "Trebuchet MS";
background:url(arkaplan.jpg) repeat-x #fff;
background-attachment: fixed;
}

#anakutu {
width: 850px;
margin: 0 auto; /* öğeyi dokümana yatay olarak ortala */
padding: 20px; /* 20 piksellik iç boşluk oranı */
background-image:url(anakutu_bg.jpg) ; /* arkaplan grafiği
yerleştir */
background-repeat: no-repeat; /* grafiği tekrar etme */
background-color: #fff; /* arkaplan rengi olarak beyaz renk
göster */
background-position: top center; /*üst ortaya konumlandırm*/
}

</style>

</head>
<body>

<div id="anakutu">
<h1> Lorem Ipsum Nedir?</h1>
<p id="ilk_paragraf">
```

Lorem Ipsum, masaüstü yayıncılık ve basın yayın sektöründe kullanılan taklit yazı bloğu olarak tanımlanır. **Lipsum**, oluşturulacak şablon ve taslaklarda içerik yerine geçerek yazı bloğunu doldurmak için kullanılır.

</p>

<p id="ikinci_paragraf">

Lipsum, 1500'lerin başlarında bir matbaacının font model kitabı oluşturmak için, bir yazı tipi kütüphanesindeki harflerin sıralamasını bozarak yerleştirdiğinden bu yana endüstri standartı haline gelmiştir.

</p>

<p>

Lorem Ipsum, 500 yıl boyunca varlığını sürdürmekle kalmamış ve günümüzde elektronik yazı tipinin gerektiği birçok konuda hazır bir araç olarak kullanılmaya başlanmıştır.<**em**> **Lipsum** 1960'larda içinde **Lorem Ipsum** paragraflarının bulunduğu letasetlerin piyasaya çıkması ve 90 larda **Lorem Ipsum** versiyonlarını içeren Aldus Pagemaker gibi programlarla beraber yaygın hale gelmiştir.

</p>

<p>

Lorem Ipsum, 500 yıl boyunca varlığını sürdürmekle kalmamış ve günümüzde elektronik yazı tipinin gerektiği birçok konuda hazır bir araç olarak kullanılmaya başlanmıştır.<**em**> **Lipsum** 1960'larda içinde **Lorem Ipsum** paragraflarının bulunduğu letasetlerin piyasaya çıkması ve 90 larda **Lorem Ipsum** versiyonlarını içeren Aldus Pagemaker gibi programlarla beraber yaygın hale gelmiştir.

</p>

<p>

Lorem Ipsum, 500 yıl boyunca varlığını sürdürmekle kalmamış ve günümüzde elektronik yazı tipinin gerektiği birçok konuda hazır bir araç olarak kullanılmaya başlanmıştır.<**em**> **Lipsum** 1960'larda içinde **Lorem Ipsum** paragraflarının bulunduğu letasetlerin piyasaya çıkması ve 90 larda **Lorem Ipsum** versiyonlarını içeren Aldus Pagemaker gibi programlarla beraber yaygın hale gelmiştir.

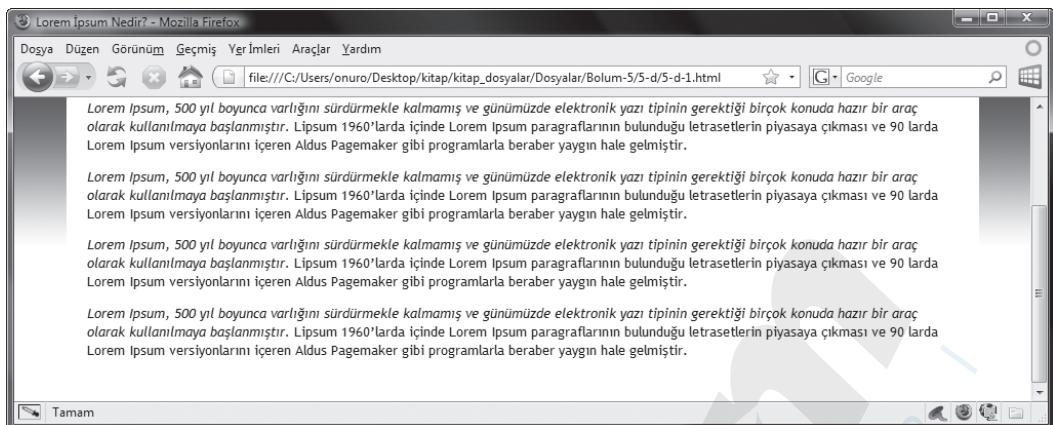
</p>

</div> <!-- #anakutu kapanışı -->

</body>

</html>

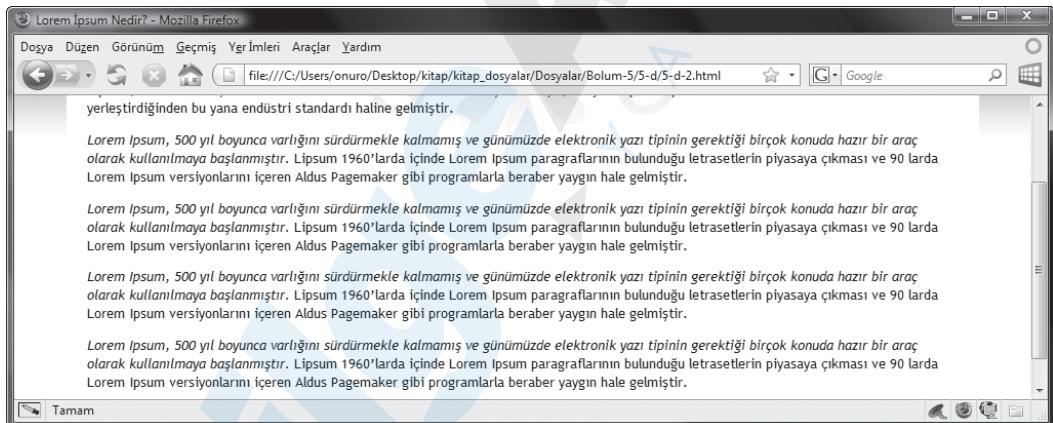
Şimdi dokümanı tarayıcıda görüntüleyip biraz aşağı scroll edecek olursak, body arkaplanının sabit kaldığını, scroll esnasında kaybolmadığını görürüz:



Figür 5.d.1

Normal haline getirmek, yani scroll haline döndürmek istersek de son eklediğimiz background-attachment özelliğini silmemiz (bu özellik girilmediği sürece varsayılan olarak scroll edecektir) veya ona scroll değeri verebiliriz:

```
body {
    font: 9.5pt "Trebuchet MS";
    background:url(arkaplan.jpg) repeat-x #fff;
    background-attachment: scroll;
}
```



Figür 5.d.2

Arkaplanın da içerkile beraber kaydığını görüyoruz. background-attachment özelliğini girmeden seydik de durum aynı olacaktı.



6 CSS Konumlandırma 1: Float

6 CSS Konumlandırma 1: Float

- Kutuları (Block seviyesi elementler) Sola ve Sağa Yaslamak
- Diğer Öğeleri Float Ettirmek

CSS Konumlandırma 1: Float

Şu ana kadar geldiğimiz noktada, temel CSS işlevlerini ve dokümanın tasarımsal özelliklerinin biçimlendirilmesi üzerine eğildik.

Bundan sonraki aşamada biraz daha ileri bir seviyeye, dokümanın görsel hiyerarşisi ve kutu yapısının konumlandırılmasıyla ilgili biçimlendirmeler üzerinde duracağız.

CSS ile öğeleri sola, sağa, aşağı, yukarı ve ortaya konumlandırmamanın son derece basit olduğunu söyleyemeyeceğim. Ancak fikri kavradıktan sonra sıkıntı çekeceğinizi sanmıyorum.

Eski tablolu tasarımını kullanmış olanların kolay bir şekilde uyguladığı (ve aynı zamanda dokümanı şişiren) konumlandırma ve hizalamalar CSS ile daha hassas bir şekilde uygulanmak durumunda. Ancak bu şekilde tüm tarayıcılarda ve farklı platform erişimlerinde Web dokümanlarımız kusursuz görünecektir.

CSS ile konumlandırma yöntemlerinin en basiti olan floating (yüzdürme) ile başlayacağız.

Float eskiden içeriğimizdeki görselleri (``) metin yanına yerleşmeleri için sola veya sağa yaslama amaçlı kullanılıyordu.

CSS'in yaygınlaşmasıyla beraber float özelliği ana kutusal elementlere de (**block level**/div, p, table vs) uygulanmaya başladı ve float'un ana öğe konumlandırmada da oldukça yardımcı olduğu fark edildi.

Bununla beraber float'un doğrudan kullanımı ile konumlandırma sorununu kökten çözen bir özellik olduğunu söylemek mümkün değildir.

Zira alıştırmalarımızda da göreceğimiz üzere float bazı ufak handikapları da beraberinde getiriyor. Konumlandırmamızı uygularken bu handikaplardan nasıl kurtulacağımızı da göreceğiz.

Float yine de tasarımını dokümana uygulama esnasında temel konumlandırma işlemlerini gerçekleştirmeye amaçlı olarak en sık kullanılan özelliklerden biridir.

Size bu özelliğin kullanımıyla ilgili güzel bir örneği yeniden sunmak istiyorum. Evvelce renklendirme bölümünde örneklediğim Tweetapp (www.tweetapp.com) sitesinin help sayfalarının içinde bulunan ve **float eden** öğelerin ekran görüntüsüne bakalım:



Figür 6

Kırmızı çerçeve içine alınan alanlar bu doküman içindeki float uygulanarak hizalanmış öğeleri ve kutuları gösteriyor.

Şimdi dilerseniz float özelliğini nerede ve nasıl kullanabiliriz, bununla ilgili uygulamalara geçelim.

6.a Kutuları (Block Seviyesi Elementler) Sola ve Sağa Yaslamak

Az önce belirttiğim gibi, float'u eski zamanlarda yalnızca dokümanımızdaki görselleri sola veya sağa yaslayarak metinlerin onların yanına yerleşmesini sağlamak amaçlı uyguluyorduk.

Şimdi göreceğimiz uygulamalarda aynı etkiyi doküman içindeki kutu elementleri yaslamak için float'u kullanıyor olacağız.

Burada bir konuya dikkat çekmek istiyorum. float özelliği hiçbir şekilde net hizalama anlamına gelmiyor. Türkçe tam karşılığı “**havada süzülmek**” olan float, öğelerimizi soyut olarak sol veya sağa yaslamaya sağlayarak konumlanmış etkisini vermeye yarayacaktır. Kesin konumlandırma amaçlı uygulamalar için ileride position özelliğini göreceğiz.

Selektör ve elementlere uygulamak istediğimiz float özelliği, üç değer alabilir:

- **left:** sola yasla
- **right:** sağa yasla
- **none:** float yok / ögeye evvelce uygulanan bir float varsa onu kaldır. Normal şartlarda float özelliği verilmemişse öğelerin standart float'u bu değere sahiptir.

Şimdi, içeriğinde sola ve sağa yaslanması gereken iki div kutusunun bulunduğu ve anakutu div'i tarafından kapsanan bir HTML dokümanı oluşturalım:

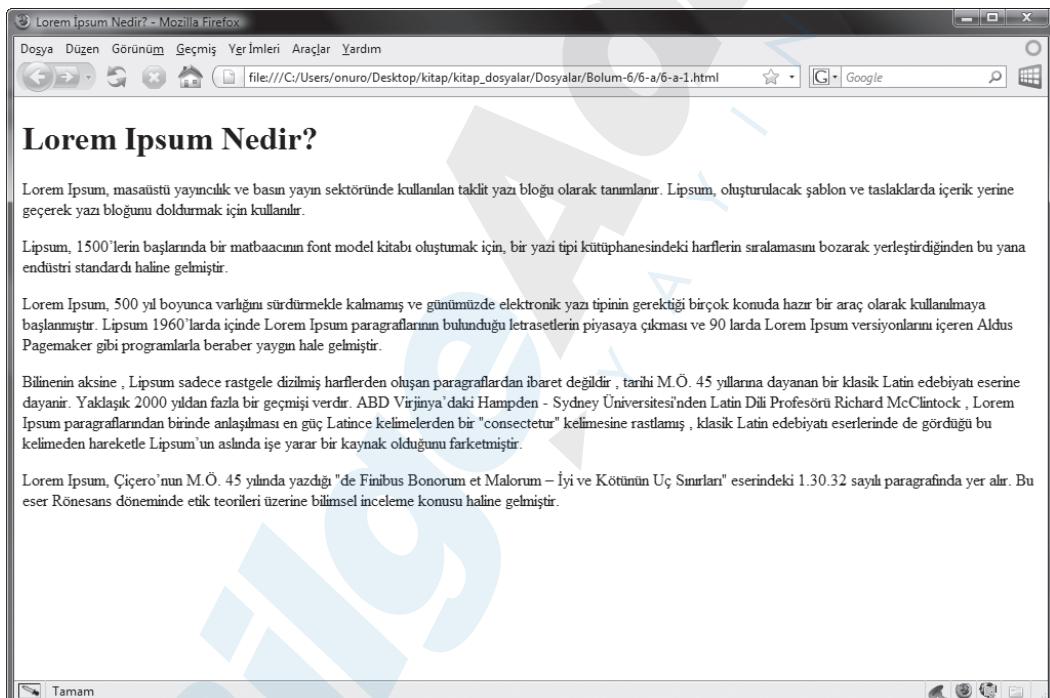
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  
<html xmlns= "http://www.w3.org/1999/xhtml">  
<head>  
<meta http-equiv="Content-Type" content="text/html" charset="utf-8"/>  
<title>Lorem Ipsum Nedir?</title>  
<style type="text/css">  
/*az sonra buraya css kodlarımızı gireceğiz*/  
</style>  
  
</head>  
<body>  
  
<div id="anakutu">  
<h1> Lorem Ipsum Nedir?</h1>  
<div id="sol_kutu">  
  
<p>  
Lorem Ipsum, masaüstü yayıncılık ve basın yayın sektöründe  
kullanılan taklit yazıbloğu olarak tanımlanır. Ipsum,  
oluşturulacak şablon ve taslaklarda içerik yerine geçerek yazı  
bloğunu doldurmak için kullanılır.  
</p>  
  
<p>  
Ipsum, 1500'lerin başlarında bir matbaacının font model kitabı  
oluşturmak için, bir yazı tipi kütüphanesindeki harflerin  
sıralamasını bozarak yerleştirdiğinden bu yana endüstri standarı  
haline gelmiştir.  
</p>  
  
<p>  
Lorem Ipsum, 500 yıl boyunca varlığını sürdürmekle kalmamış ve  
günümüzde elektronik yazıtipinin gerektiği birçok konuda hazır  
bir araç olarak kullanılmaya başlanmıştır. Ipsum 1960'larda  
içinde Lorem Ipsum paragraflarının bulunduğu letasetlerin  
piyasaya çıkması ve 90'larda Lorem Ipsum versiyonlarını içeren  
Aldus Pagemaker gibi programlarla beraber yaygın hale gelmiştir.  
</p>  
  
</div><!-- #sol_kutu kapanışı -->  
<div id="sag_kutu">  
  
<p>  
Bilinenin aksine, Ipsum sadece rastgele dizilmiş harflerden  
oluşan paragraflardan ibaret değildir, tarihi M.Ö. 45 yıllarına  
dayanan bir klasik Latin edebiyatı eserine dayanır. Yaklaşık 2000  
yıldan fazla bir geçmişi vardır. ABD Virjinya'daki Hampden -  
Sydney Üniversitesi'nden Latin Dili Profesörü Richard McClintonck
```

, **, Lorem Ipsum paragraflarından birinde anlaşılması en güç Latince kelimelerden bir "consectetur" kelimesine rastlamış , klasik Latin edebiyatı eserlerinde de gördüğü bu kelimededen hareketle Lipsum'un aslında işe yarar bir kaynak olduğunu farketmiştir.</p>**

```
<p>
  Lorem Ipsum, Çiçero'nun M.Ö. 45 yılında yazdığı "de Finibus Bonorum et Malorum – İyi ve Kötünün Uç Sınırları" eserindeki 1.30.32 sayılı paragrafında yer alır. Bu eser Rönesans döneminde etik teorileri üzerine bilimsel inceleme konusu haline gelmiştir.
</p>

</div> <!-- #sag_kutu kapanışı -->
</div> <!-- #anakutu kapanışı -->

</body>
</html>
```



Figür 6.a.1

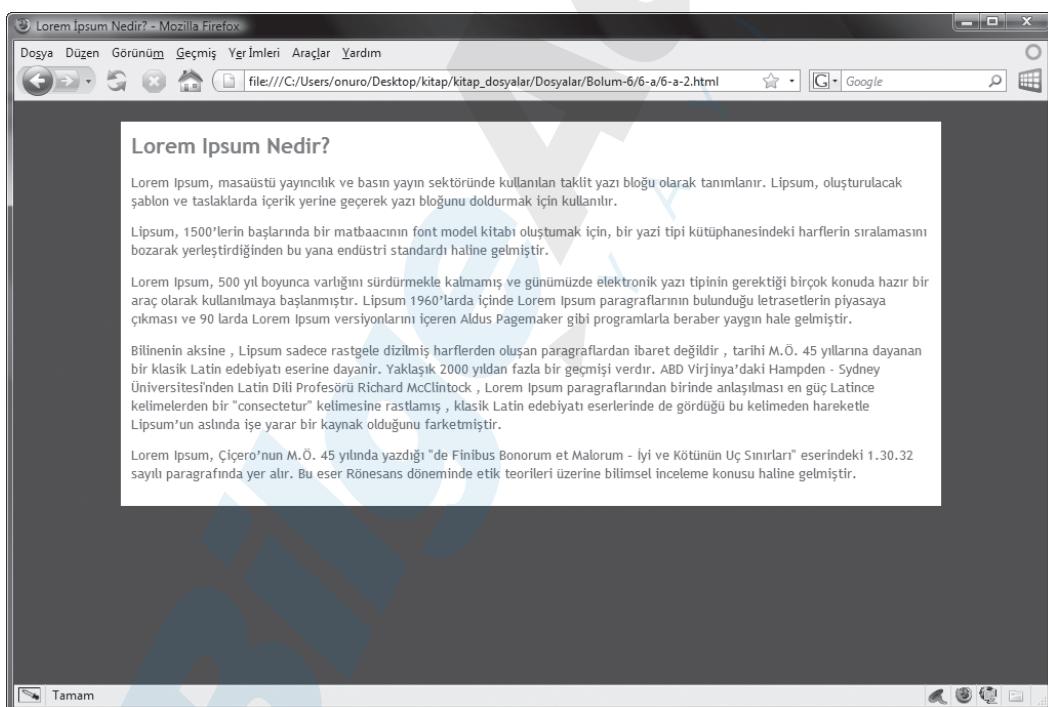
Sol ve sağ kutunun stillendirmesine geçmeden önce dokümanı biraz daha albenili hale getirelim:

```
body {
  /* doküman geneli stillendirme */
  background:#503b28; /* koyu pastel kahverengi arkplan rengi */
  color:#666; /* koyu gri metin rengi */
  font:9.5pt "Trebuchet MS";
  text-align:center; /* tüm içeriği ortala (Internet Explorer ortalama problemi için) */
  margin:20px; /* doküman geneli boşluk oranı */
```

```

}
#anakutu {
width:780px;
background:#fff;
padding:10px; /* eklenen padding değerleri nedeniyle anakutu
genişliği aslında 800px */
margin:0 auto; /* kutuyu doküman geneline ortalı */
text-align:left; /* Internet Explorer için uygulanan center
konumunu düzelt */
}
#anakutu h1 {
color:#8c7760; /* pastel kahverengi metin rengi */
margin:0; /* h1 elementinin standart margin boşluğunu sıfırla */
padding:0; /* h1 elementinin standart padding boşluğunu sıfırla
*/
font-size:16pt; /* h1 elementinin standart yazıtılı boyutunu
16 pointa sabitle */
}

```

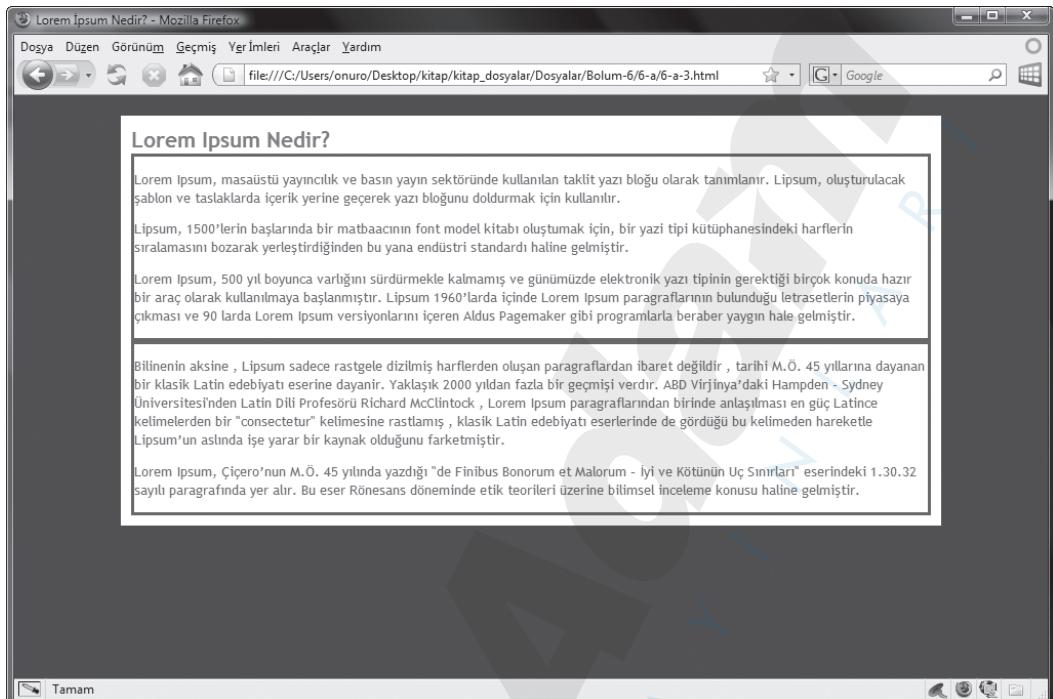


Figür 6.a.2

Tüm tarayıcılarda neredeyse aynı ortak görünümü sahip olan genel stillendirmeyi uygulamış olduk. (Internet Explorer sonucuna da bakacak olursak yalnızca paragraf boşluklarının [margin] fark yarattığını görürüz. Onu şimdilik öylece bırakalım.)

Şimdi dilerseniz sol ve sağ kutuya yaslama uygulamadan önce, onların kapsama alanlarını daha iyi görmek için kendilerine çerçeve vererek biraz belirginştirelim:

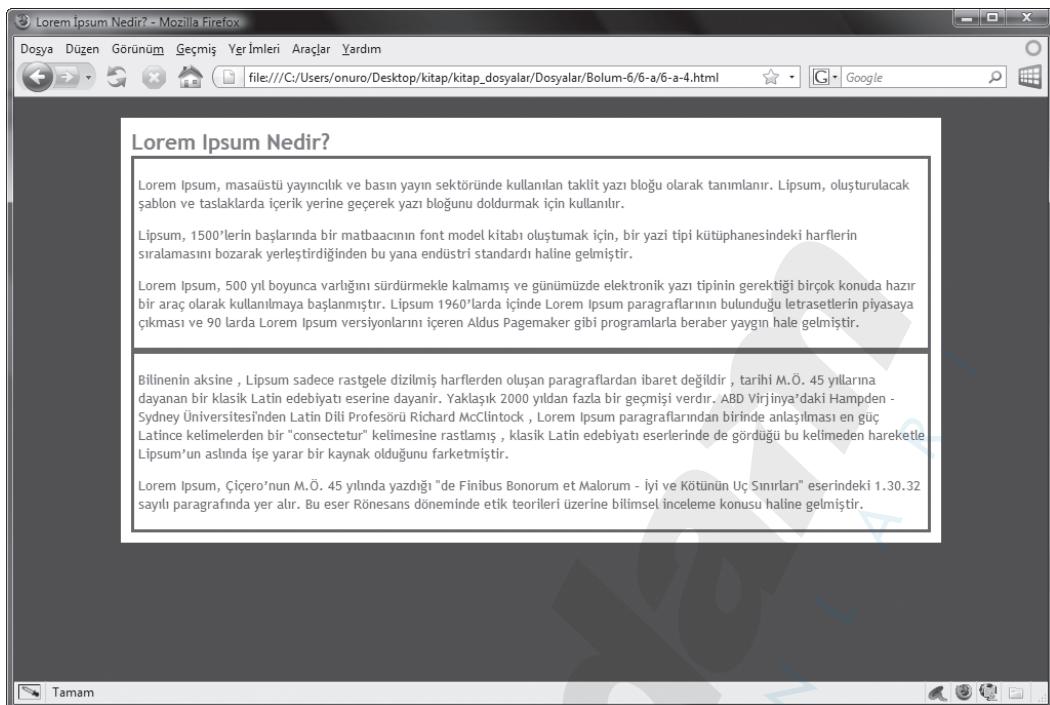
```
#sol_kutu, #sag_kutu {
/*selektör grublama ile iki kutuya da ortak özellikler */
border:3px solid #685a4b; /* 3 piksellik kahverengi tek ton
çerçeve */
}
```



Figür 6.a.3

Sol ve sağ kutunun kapsam alanlarını belirginleştirmiş olduk. Yalnız kutuların içindeki metinler boşluksuz olarak göze batıyor gibi. Kutulara iç boşluk (padding) ekleyerek okunaklığını biraz artıralım:

```
#sol_kutu, #sag_kutu {
/*selektör grublama ile iki kutuya da ortak özellikler */
border:3px solid #685a4b; /* 3 piksellik kahverengi tek ton
çerçeve */
padding:3.5px;
}
```

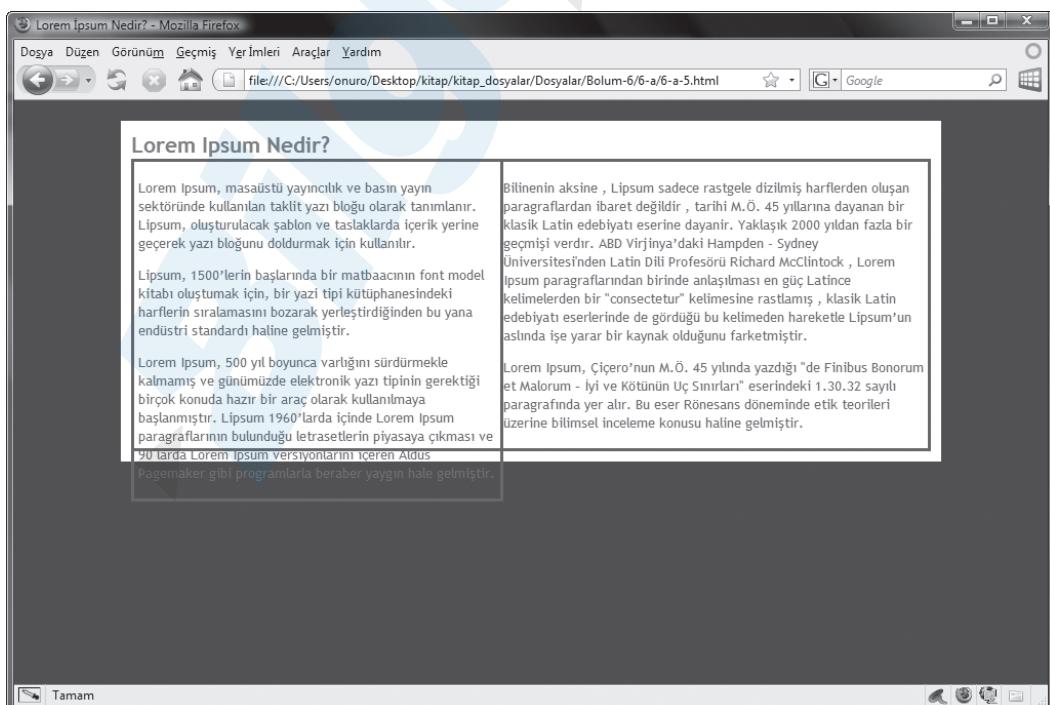


Figür 6.a.4

Artık kutuları hizalamak için hazırız.

Öncelikle `sol_kutu` div'imize anakutumuzun yarı genişliğine yakın bir oranda genişlik verdikten sonra sola float ettirerek yaslayalım:

```
#sol_kutu {
    width:370px;
    float: left;
}
```



Figür 6.a.5

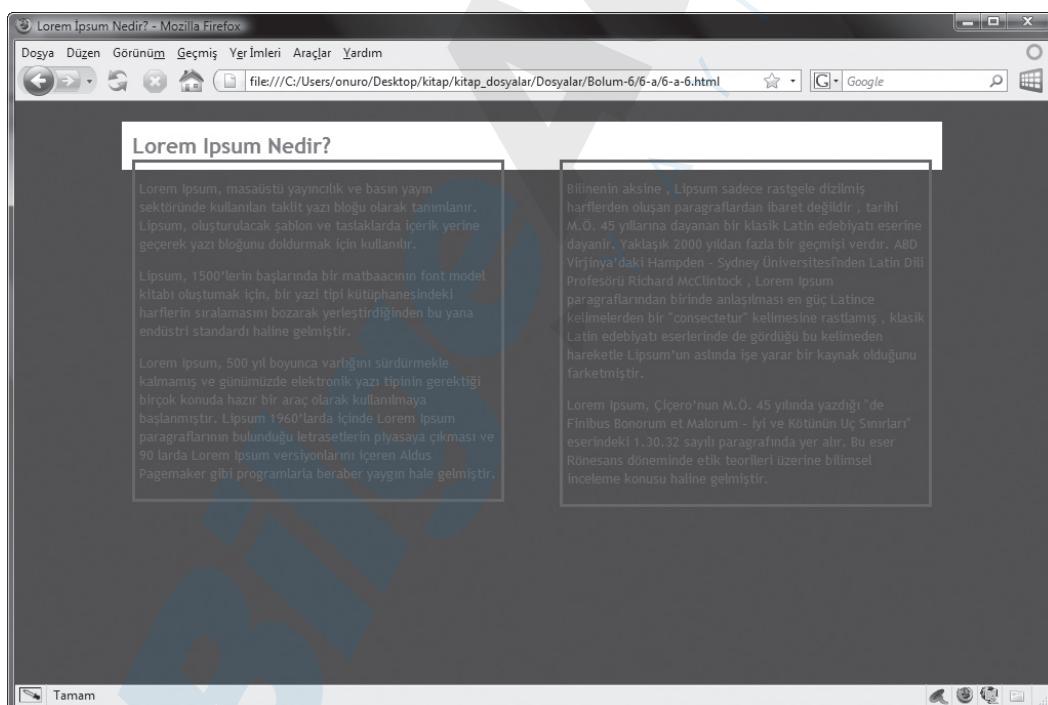
Eyvah, her şey birbirine girdi!

Aslında endişe edecek bir şey yok. Burada sol kutuya yalnızca float uygulamamızdan ötürü neler gerçekleştiğinden biraz bahsedeyim. Zira float özelliğinin kilit noktaları aşağıda bahsettiğim durumlar ve bunların çözümleridir:

- **Float eden bir veya daha fazla öğe, kendilerinin bir üst elementinin yüksekliğini kıralar.** Bizim örneğimizde sol kutunun bir üst elementi anakutudur ve onun yüksekliği aslında kırıldı. Halen anakutu ve arkaplan beyazını görebilmemizin sebebi, henüz sağ kutuya float uygulamamış olmamızdır.
- **370 piksellik alan boyunca devam eden sol kutuya uygulanan yaslama sağ kutunun da solun hemen yanına yapışmasına neden oldu.** Bu da float'un doğasıdır. Eğer sol kutu yerine bir görsel () koyup aynı özellikleri uygulamış olsaydık da bu sonucu elde edecektik, değil mi?

Bilginiz olsun, her float uyguladığımızda bu sıkıntılarla karşılaşacağız. Bu sıkıntıları çözmeden önce sağ kutunun da yaslamasını yaparak sonuca bir bakalım:

```
#sag_kutu {
    width: 370px;
    float: right;
}
```



Figür 6.a.6

Anakutu yüksekliğinin kırılmasını göz ardı edecek olursak, yapmak istediğimiz şeyi gerçekleştirmiş olduk.

Peki, anakutu div'imizin yüksekliğini nasıl kurtarabilirim?

Burada mantık yürütecek olursak, anakutuya sabit bir yükseklik (height) değeri vererek sorunu kökünden halledebileceğimizi düşünebiliriz. Peki, ya içeriğimiz ileride daha da artarsa, sayfamız dinamikse ve içeriklerin değişken yoğunluğu nedeniyle doküman yüksekliği (scroll oranı) devamlı değişiyorsa? Bilgisayarımızın başında oturup günde 7-8 defa anakutu div'inin yükseklik değerini düzeltmekle mi uğraşacağız?

Anakutuya yükseklik uygulamak, kesinlikle mantıklı ve kökten çözüm olmayacak. Başka bir şeye ihtiyacımız var. Bu tür durumlardan bizi kurtaran bir özellik gerekiyor.

Nitekim CSS içinde böyle bir özellik mevcuttur.

Float Eden Öğeleri Temizlemek / Kurtarıcı Kahramanlar (Clear Özelliği)

Float eden öğeler devamlı bize yukarıdaki sıkıntıları çıkarmaya devam edeceği için, bu tür durumlarda devreye bir tür CSS hacking teknigi devreye girecek.

Yapacağımız işlemler sırasıyla şu şekilde:

- Kendi belirleyeceğimiz bir adla class selektöre sahip bir div oluşturacağız. Ben isimlendirme yaparken genelde “**kurtarıcı**” ismini kullanıyorum. Selektör ürünün class olmasının sebebi şudur: Bir dokümanda bu tip kurtarmalara birden fazla kez ihtiyacımız olabilir. Class selektörler dokümanda tekrar tekrar çağrılabildikleri için en doğru seçim olacaktır. Id selektörleri tercih etmeye kalkarsak, bu selektör türü dokümanda yalnızca bir kez kullanılabiligidinden, işimizi görmeyeceklerdi.
- Gereken her noktada kurtarıcı div kutusunu HTML dokümanımızda float eden son ögeden hemen sonra yerleştiriyor olacağız
- Kurtarıcı class selektörünün özellikleri aşağıdaki gibi olacaktır:
 - **Clear** özelliğine sol, sağ veya ikisi birden değeri verildiğinde float yaslamaları ortadan kalkacak.
 - Div içeriğinde hiçbir şey bulunmayacağı için, kendisinin **yüksekliği** 0px değerine sahip olacak.
 - Div yüksekliğinin tüm tarayıcılarda 0 olduğundan emin olmak için **font boyutu** da 0px oranına indirgenecek.
 - Kurtarıcı div bir nevi gizli kahraman statüsüne sahip olduğu için, görünürlük (**visibility**) özelliğine gizli (**hidden**) değeri vererek tüm tarayıcılara aslında orada olmadığını ancak sıkıntıları çözüdüğünü belirteceğiz.
 - Kurtarıcı div'in iki tarafı da boydan boyaya temizlediğinden emin olmak için **genişliği** "%100" oranına sahip olacak.

Burada kilit işlemi gerçekleştirecek olan clear özelliği aşağıdaki değerleri alabilir:

- **Left:** sol tarafı temizle
- **Right:** sağ tarafı temizle
- **Both:** hem sağ, hem sol, iki tarafı da temizle

Dolayısıyla clear özelliğinin asıl maksadının etrafta float eden öğeler varsa, onlardan sonra gelecek içeriğin float eden öğelerin yanına yaslanmadan normal akışına devam etmelerini sağlayan bir işleve sahip olduğunu söyleyebiliriz.

Şimdi dilerseniz öncelikle içi boş kurtarıcı sınıfına sahip bir div oluşturalım ve bunu float eden kutularımızdan hemen sonra yerleştirelim:

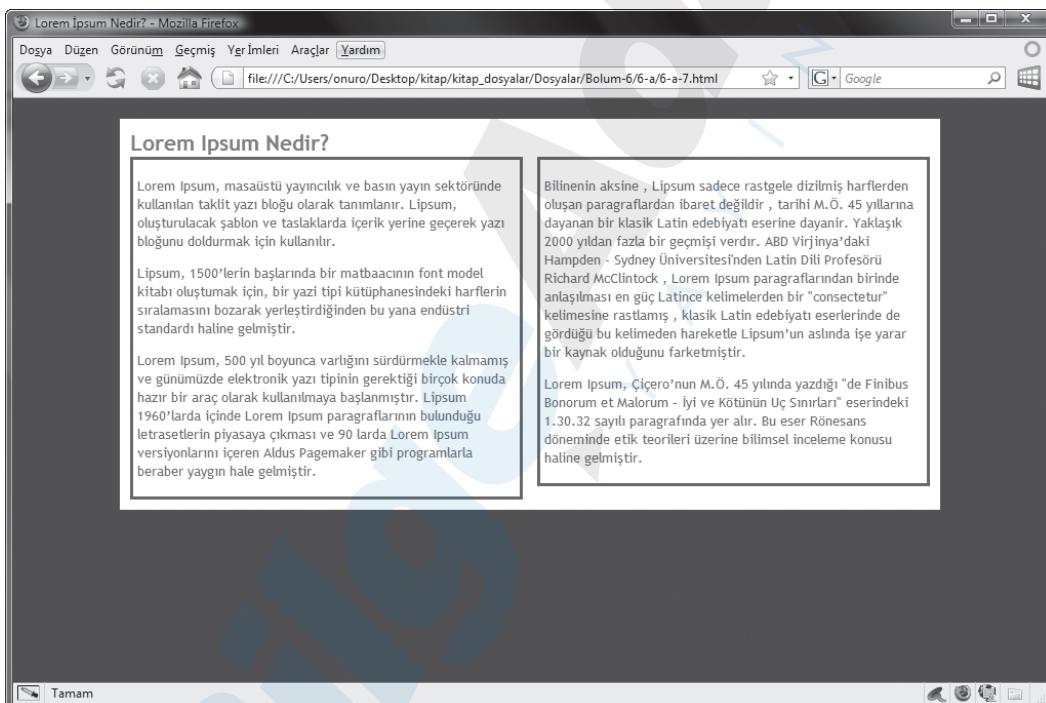
```
...
Rönesans döneminde etik teorileri üzerine bilimsel inceleme
konusu haline gelmiştir.
</p>
</div> <!-- #sag_kutu kapanışı / float eden son öge-->
<div class="kurtarici">&nbsp;</div>
</div> <!-- #anakutu kapanışı -->
</body>
</html>
```

Kurtarıcı sınıfına sahip div'imize mutlaka en son float eden öğeden sonra yerleştirmemiz gerektiğini hatırlatmak istiyorum. Kurtarıcı div'in yanlış noktalara yerleştirilmesi, farklı görünümlere ve sıkıntılı hizalamalara neden olabilir.

xHTML konusunda bilgi sahibi olanlarınız zaten farkındadır, ancak ben yine de belirteyim: Kurtarıcı div'in içeriğinin boşmasına rağmen yerleştirdiğimiz boşluk () özel karakterinin sebebi, xHTML dokümanları içinde içi boş etiket açılamama kuralıdır.

Şimdi, evvelce bahsettiğim kurtarıcı selektörünün özelliklerini girip değerlerini atayalım:

```
.kurtarici {
    clear: both;
    height: 0px;
    font-size: 0px;
    visibility: hidden;
    width: 100%;
}
```



Figür 6.a.7

İşte bu kadar. Anakutunun değişken yüksekliği geri geldi ve içeriğinde sıkıntı yaratmadan duran sol ve sağa yaslı iki kutuya sahip. Bütün bunları mümkün kılan şey kurtarıcı div'ımız ve clear özelliğidir.

Kurtarıcı div'ımızı gereken her noktada kullanmaya devam edeceğiz.

6.b Diğer Öğeleri Float Ettirmek

Daha önce de ifade ettiğim gibi, float dilediğimiz her elemente uygulanabilir. Tercih ettiğimiz noktalarda float uygulayarak dokümanımızın görünümüne yeni açılar kazandırabiliriz.

Bu kez dokümanımızdaki paragrafların tamamına float uygulayarak yan yana dizili kutular görünen elde etmeye çalışalım.

Bu uygulama için öncelikle içeriğinde birkaç paragraf bulunan bir HTML dokümanı oluşturup biraz stillendirme uyguluyoruz:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns= "http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type " content="text/html" charset="utf-8"/>
<title>Lorem Ipsum Nedir?</title>
<style type="text/css">
body {
/* doküman geneli stillendirme */
background:#503b28; /* koyu pastel kahverengi arkplan rengi */
color:#666; /* koyu gri metin rengi */
font:9.5pt "Trebuchet MS";
text-align:center; /* tüm içeriği ortala (Internet Explorer
ortalama problemi için) */
margin:20px; /* doküman geneli boşluk oranı */
}
#anakutu {
width:930px;
background:#fff;
padding:10px; /* eklenen padding değerleri nedeniyle anakutu
genişliği aslında 950px */
margin:0 auto; /* kutuyu doküman geneline ortala */
text-align:left; /* Internet Explorer için uygulanan center
konumunu düzelt */
}
#anakutu h1 {
color:#8c7760; /* pastel kahverengi metin rengi */
margin:0; /* h1 elementinin standart margin boşluğunu sıfırla */
padding:0; /* h1 elementinin standart padding boşluğunu sıfırla */
font-size:16pt; /* h1 elementinin standart yazıtipi boyutunu
16 puntoya sabitle */
}
</style>
</head>
<body>
<div id="anakutu">
<h1> Lorem Ipsum Nedir?</h1>
<p>
Lorem Ipsum, masaüstü yayıncılık ve basın yayın sektöründe
kullanılan taklit yazı blogu olarak tanımlanır. Lipsum,
oluşturulacak şablon ve taslaklarda içerik yerine geçerek yazı
bloğunu doldurmak için kullanılır.
</p>
```

```
<p>
Lipsum, 1500'lerin başlarında bir matbaacının font model kitabı
oluştumak için, bir yazı tipi kütüphanesindeki harflerin
sıralamasını bozarak yerleştirdiğinden bu yana endüstri standarı
haline gelmiştir.
</p>

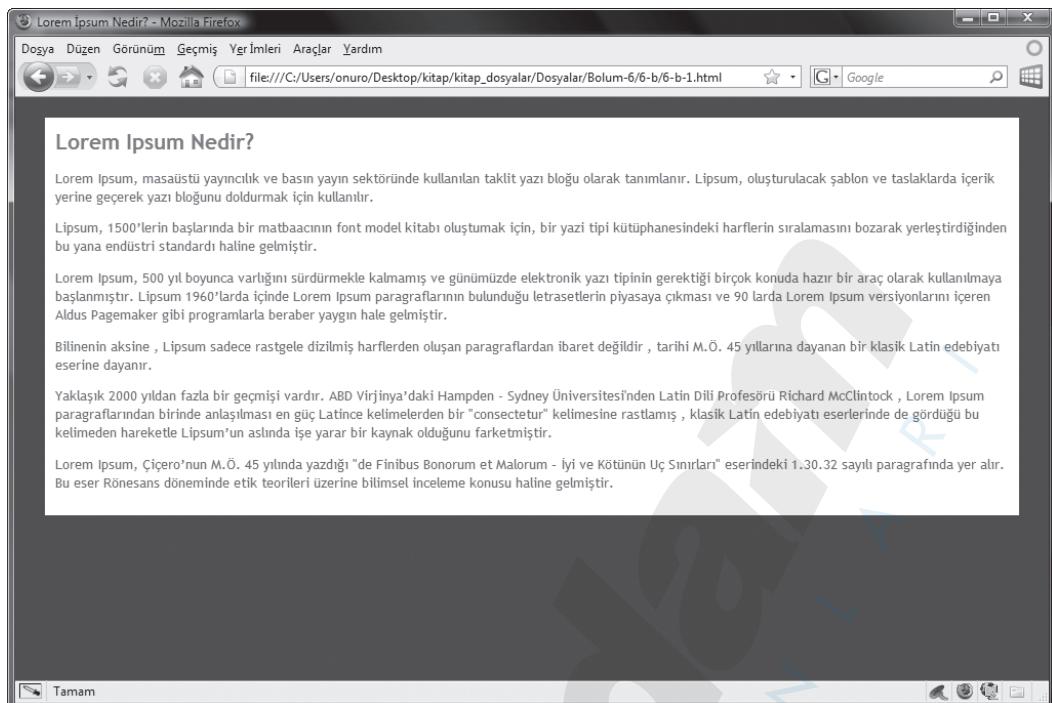
<p>
Lorem Ipsum, 500 yıl boyunca varlığını sürdürmekle kalmamış ve
günümüzde elektronik yazı tipinin gereği birçok konuda hazır
bir araç olarak kullanılmaya başlanmıştır. Lipsum 1960'larda
içinde Lorem Ipsum paragraflarının bulunduğu letasetlerin
piyasaya çıkması ve 90 larda Lorem Ipsum versiyonlarını içeren
Aldus Pagemaker gibi programlarla beraber yaygın hale gelmiştir.
</p>

<p>
Bilinenin aksine, Lipsum sadece rastgele dizilmiş harflerden
oluşan paragraflardan ibaret değildir, tarihi M.Ö. 45 yıllarına
dayanan bir klasik Latin edebiyatı eserine dayanır.
</p>

<p>
Yaklaşık 2000 yıldan fazla bir geçmişi vardır. ABD Virjinya'daki
Hampden - Sydney Üniversitesi'nden Latin Dili Profesörü Richard
McClintock, Lorem Ipsum paragraflarından birinde anlaşılması en
güç Latinçe kelimelerden bir "consectetur" kelimesine rastlamış
, klasik Latin edebiyatı eserlerinde de gördüğü bu kelimededen
hareketle Lipsum'un aslında işe yarar bir kaynak olduğunu
farketmiştir.
</p>

<p>
Lorem Ipsum, Çiçero'nun M.Ö. 45 yılında yazdığı "de Finibus
Bonorum et Malorum – İyi ve Kötünün Üç Sınırları" eserindeki
1.30.32 sayılı paragrafında yer alır. Bu eser Rönesans döneminde
etik teorileri üzerine bilimsel inceleme konusu haline gelmiştir.
</p>

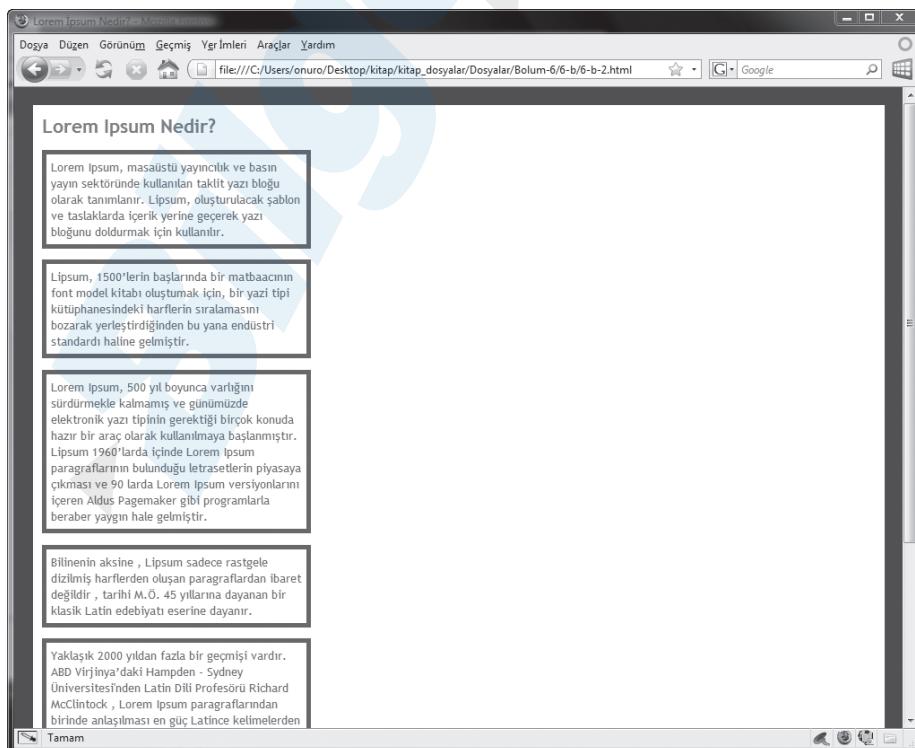
</div> <!-- #anakutu kapanışı -->
</body>
</html>
```



Figür 6.b.1

Şimdi anakutunun içinde yer alan 6 adet paragrafin hatlarını biraz belirginleştirip onlara sabit birer genişlik değeri atayalım:

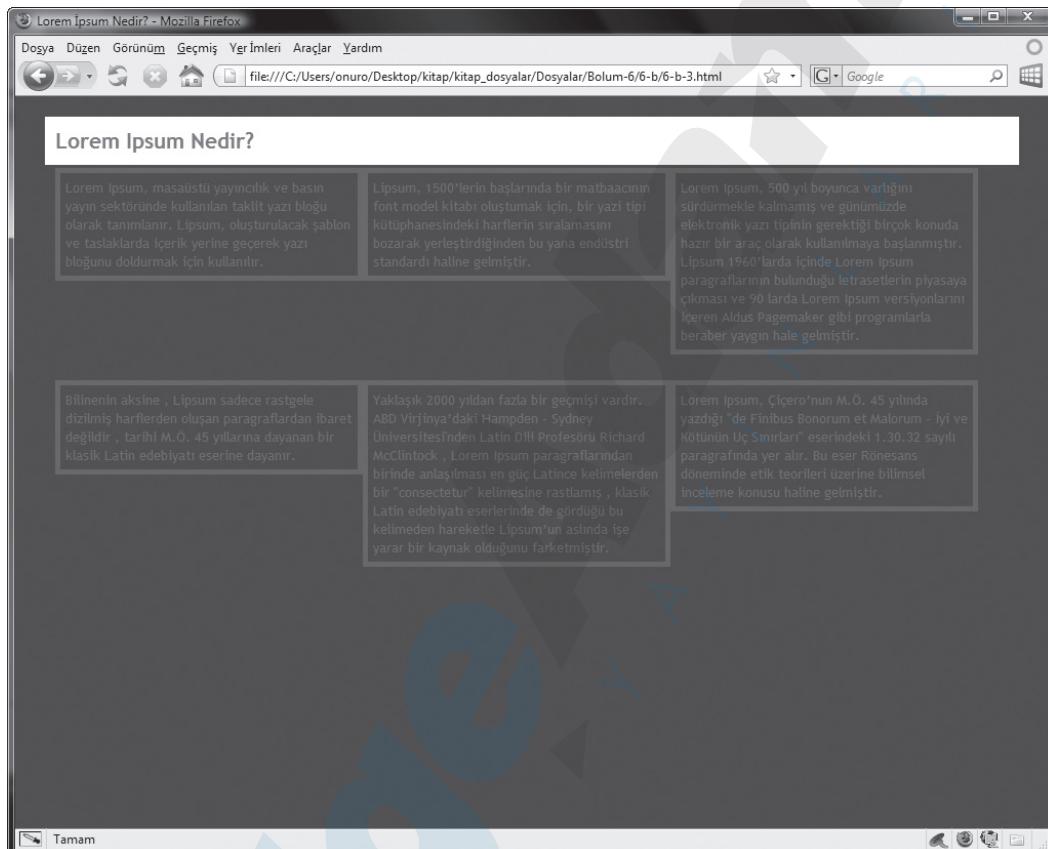
```
#anakutu p {
    width:280px;
    border: 5px solid #685a4b;
    padding:5px;
}
```



Figür 6.b.2

Sıra paragrafları sola yaslamaya geldi:

```
#anakutu p {
    width:280px;
    border: 5px solid #685a4b;
    padding:5px;
    float: left;
}
```



Figür 6.b.3

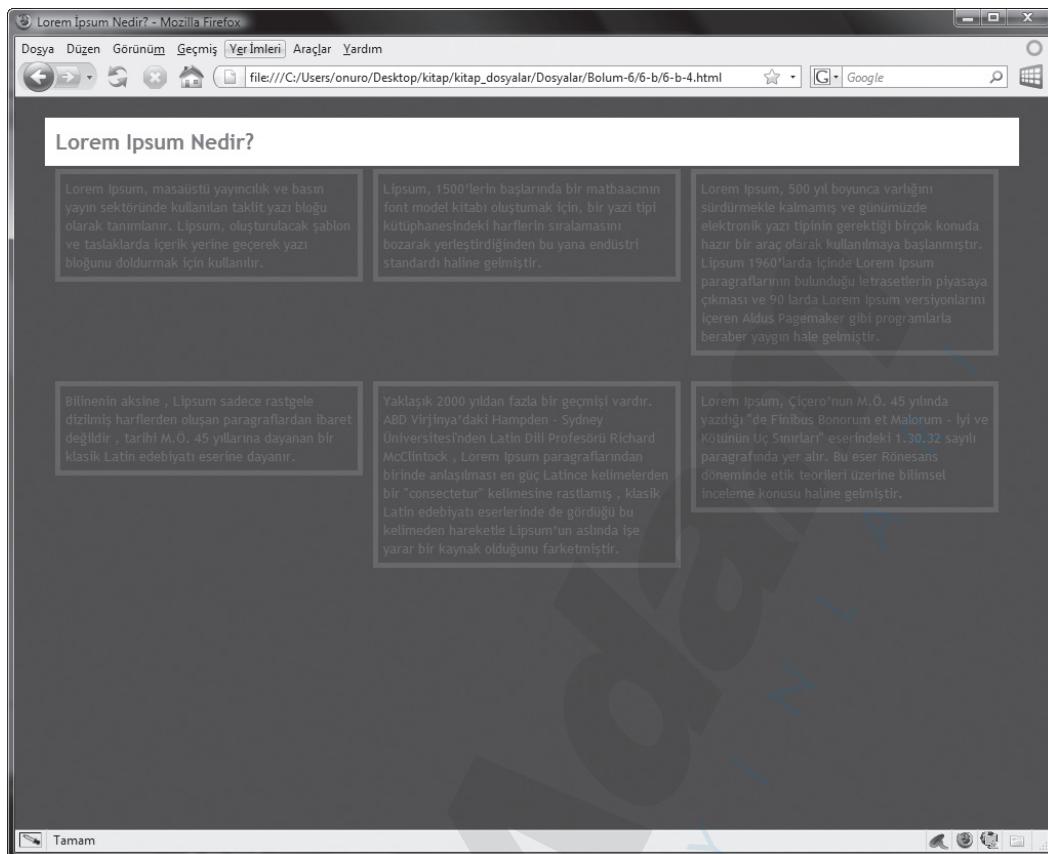
Tekrardan, anakutu div'inin yüksekliğinin kırılması problemiyle karşı karşıyayız. Bu sorunu yine bir kurtarıcı ile çözeceğiz.

Bunun dışında fark ettiyseniz öğeleri yan yana dizme işlemini float yardımıyla kolaylıkla gerçekleştirmiş olduk. Anakutu içindeki tüm paragraflara float left uyguladığımız için her paragraf birbirinin yanına yaslanıyor.

Bu şekilde birden fazla ögeyi yan yana float yardımıyla dizmek mümkün.

Şimdi dilerseniz paragrafların sağ boşluk oranını biraz daha açalım, zira şu an fazla bitişik görünyorlar:

```
#anakutu p {
    width:280px;
    border: 5px solid #685a4b;
    padding:5px;
    float: left;
    margin-right:10px; }
```



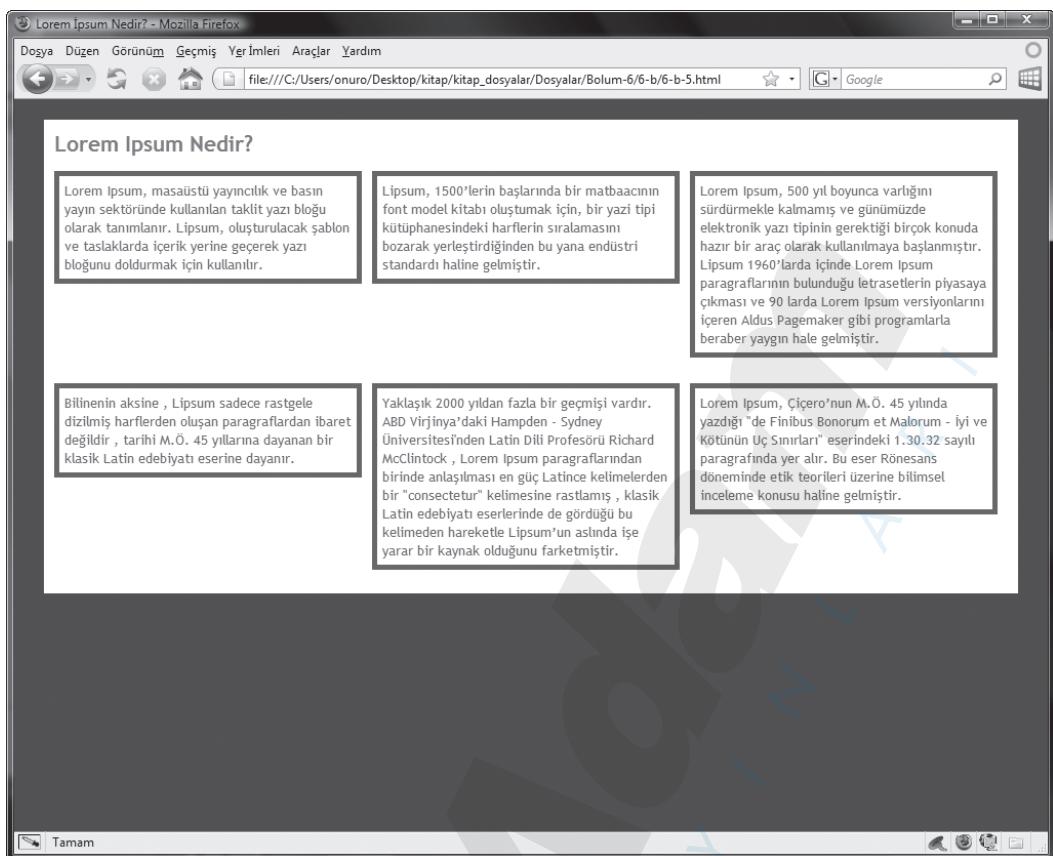
Figür 6.b.4

Artık kurtarıcıyı yerleştirebilir, anakutunun yüksekliğini kurtarabiliriz:

```
...
1.30.32 sayılı paragrafında yer alır. Bu eser Rönesans döneminde etik teorileri üzerine bilimsel inceleme konusu haline gelmiştir.
</p><!--float eden son paragraf -->
<div class="kurtarici">&nbsp;</div>
</div> <!-- #anakutu kapanışı -->
</body>
</html>
```

Kurtarıcının stillendirmesini uygulayalım:

```
.kurtarici {
    clear: both;
    height: 0px;
    font-size: 0px;
    visibility: hidden;
    width: 100%;
}
```

**Figür 6.b.5**

Anakutuya yüksekliğini geri kazandırarak, yan yana yaslanan paragraf kutuları hazırlamış olduk.

Float kullanarak hazırlayacağımız hizalama uygulamalarının sınırı yoktur. Bu konuda gelişmeye yönelik bol bol deneme ve alıştırma yapmanızı öneririm.



7

**CSS ile Site
Dolaşım
Menüsü /
Navigasyon
Oluşturmak**

7 CSS ile Site Dolaşım Menüsü / Navigasyon Oluşturmak

- HTML'in Sıralı ve Sırasız Liste Öğeleriyle Menü Oluşturma Mantığı
- Liste Öğelerindeki Rakam ve İmleçleri Kaldırmak
- Liste İmleçleri Yerine Grafikler Kullanmak
- CSS ile Metin Tabanlı Yatay Navigasyon Oluşturmak
- CSS ile Yatay Sekmeli (Tabbed) Grafik Tabanlı Navigasyon Oluşturmak

CSS ile Site Dolaşım Menüsü / Navigasyon Oluşturmak

7.a HTML'in Sıralı ve Sırasız Liste Öğeleriyle Menü Oluşturma Mantığı

CSS ile tasarımın popüler hale geldiği vakte kadar, yıllarca menülerimizi tablo içeriği olacak şekilde, şişkin kodlar ve Javascript destekli mouse efektleri yardımıyla oluşturduk.

CSS bize, çok daha kısa kodlarla esnek ve stil sahibi görünümle menüler ve site dolaşım yapıları hazırlamamıza imkan sağlıyor.

Stillendirme yaparak hazırlayabileceğimiz menülerin oluşturulmanın onlarca yolu var. Ancak ben en popüler ve erişilebilir olanını, HTML içindeki standart liste öğelerini CSS yardımıyla site dolaşım menülerine dönüştürme mucizesini anlatıyor olacağım.

7.b Liste Öğelerindeki Rakam ve İmleçleri Kaldırmak

HTML konusunda bilgi sahibi olanlarınızın belki yalnızca "yapılacaklar listesi" veya "kitap indeksi" gibi temel içerik amaçlı kullanıldığını düşünebileceğiniz liste öğeleri, CSS yardımıyla çok daha fazla anlam kazanacak.

Bilindiği üzere liste öğeleri, bir liste kapsayıcı etiket (``) ve her öğeye ait olmak üzere liste etiketlerinden (``) ibarettir.

Bu bölümde, size içinde linkler bulunan standart bir listeyi nasıl daha albenili bir dolaşım menüsü haline getirebileceğimizle ilgili bir uygulama örneği sunacağım.

Öncelikle, içinde birkaç linkin bulunduğu liste içeren bir HTML dokümanı oluşturalım:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns= "http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type " content="text/html" charset="utf-8"/>
<title>Tasarımcılar İçin İlham Kaynakları</title>
<style type="text/css">
/*az sonra buraya css kodlarını gireceğiz*/
</style>
</head>
<body>
    <h1> Tasarımcılar için İlham Kaynakları </h1>
    <ul>
        <li><a href="http://www.cssvault.com">CSS Vault</a></li>
        <li><a href=" http://www.cssbeauty.com">CSS Beauty</a></li>
        <li><a href=" http://www.bestwebgallery.com">Best Web Gallery
        </a></li>
```

```

<li><a href=" http://www.cssmania.com">CSS Mania Natural  

Feeling</a></li>

<li><a href=" http://www.thebestdesigns.com">The Best Designs  

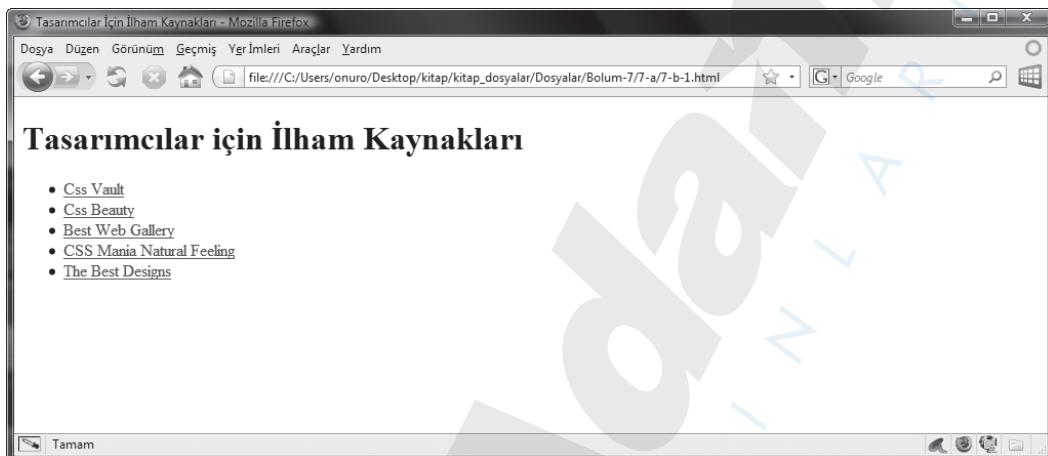
</a></li>  

</ul>

</body>  

</html>

```



Figür 7-b-1

Stilsiz olarak çok albenili görünmeyen bu link listesini ana elementine (`ul`) arkaplan grafiği atadıktan sonra liste öğeleri (`li`) ve link (`a`) özelliklerini biçimleyerek sık bir görünümü kavuşturtmaya çalışacağız.

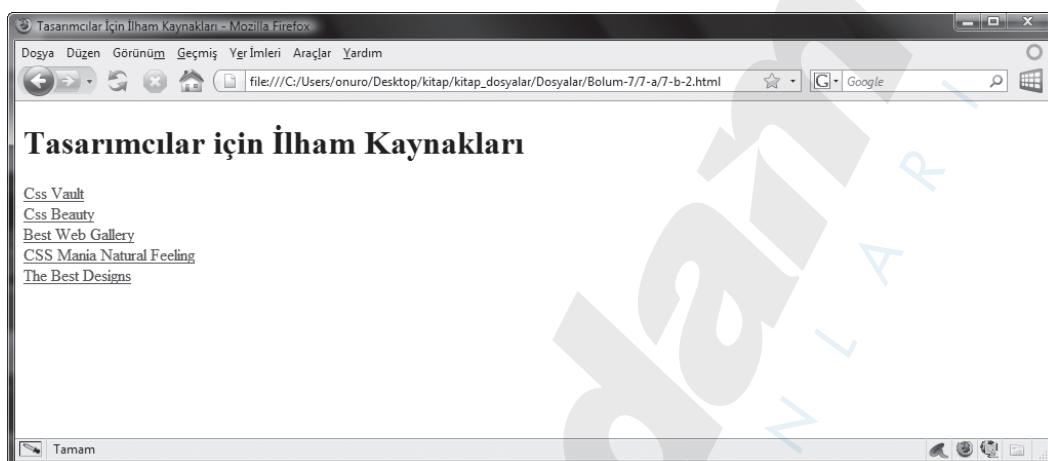
Ul elementinin arkasına yerleştireceğim grafiği Photoshop yardımıyla 202 piksel genişliğe ve 195 piksel yüksekliğe sahip olacak şekilde “liste-arkaplan.jpg” dosya adıyla kaydetmiştim:



Dolayısıyla liste öğeleri bu grafiğin bulunduğu kutu içerisinde yer alacaklar.

Grafiği `ul` elementine atamadan önce, bu elementin tarayıcı standarı stillendirmede sahip olduğu özellikler (`padding` ve `margin`) sıkıntı yaratabilir. Zira `ul` genişlik yükseklikleri boşluk oranı olmadan 202 piksele 195 piksel oranlarını hiçbir şekilde geçmemek durumunda. Ancak böylelikle muhtemel kayma ve fazla boşlukları önlleyebiliriz:

```
ul {
margin:0; /*dış boşluğu sıfırla*/
padding:0; /*iç boşluğu sıfırla*/
width: 202px; /*arkaplan grafiğinin genişliği*/
height: 195px; /*arkaplan grafiğinin yüksekliği*/
}
```



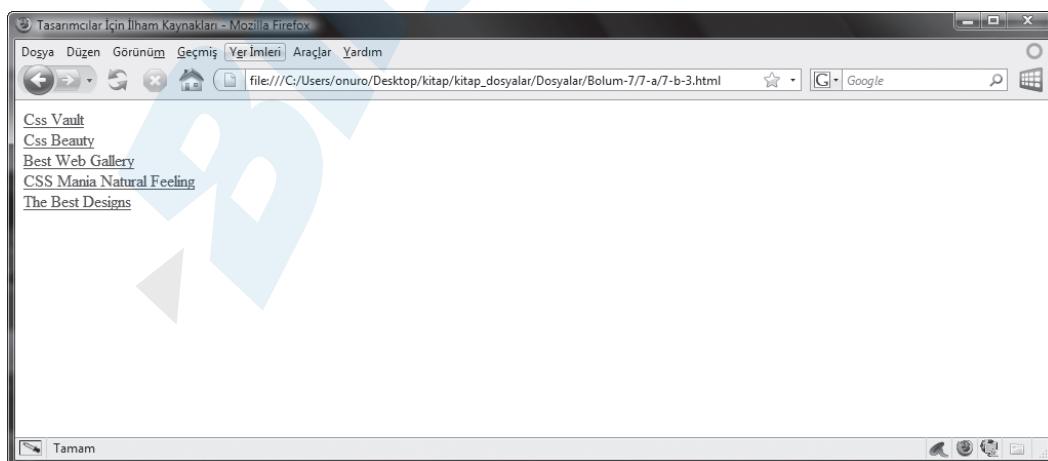
Figür 7-b-2

Boşluk oranlarını sıfırlayıp ul ana elementinin genişlik ve yükseklik değerlerini vererek arkaplan grafiğini bu elemente yerleştirmek üzere hazırlamış olduk.

Grafiği arkaplan olarak atmadan önce yapılması gereken bir şey daha var. Fark ettiyseniz h1 başlığımızın içeriği arkaplan grafiğimiz içindeki metinde zaten yer alıyor. Bu durumda iki defa tekrar eden başlığa sahip olmak yerine CSS yüklü görünümde h1 elementimizi ortadan kaldırabilir, stilsiz görüntüleyen ziyaretçilerin tekrar bu başlığı görmesini sağlayabiliriz.

Bu nedenle h1 elementini ortadan kaldırıralım:

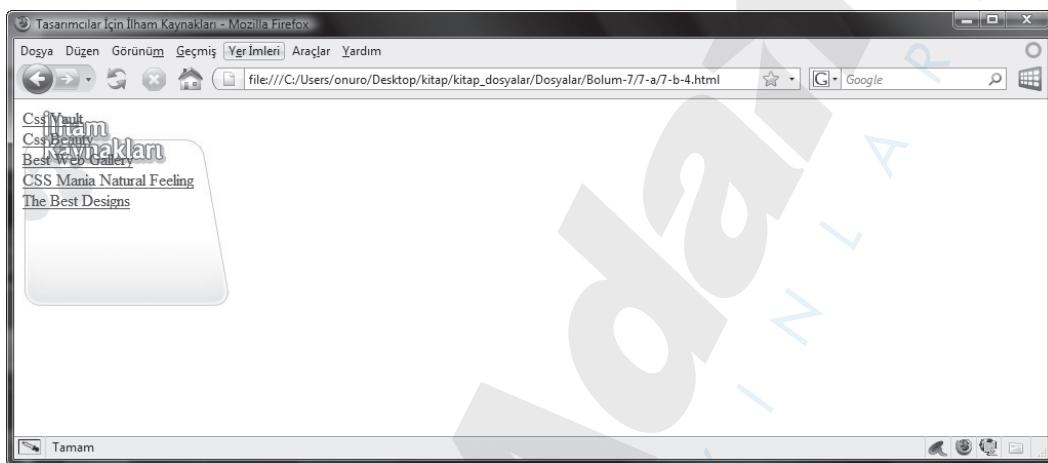
```
h1 {
display:none;
}
```



Figür 7-b-3

Şimdi arkaplanı ul içine atamaya hazırız:

```
ul {
margin:0; /*dış boşluğu sıfırla*/
padding:0; /*iç boşluğu sıfırla*/
width: 202px; /*arkaplan grafiğinin genişliği*/
height: 195px; /*arkaplan grafiğinin yüksekliği*/
background: url(liste-arkaplan.jpg) no-repeat;
}
```

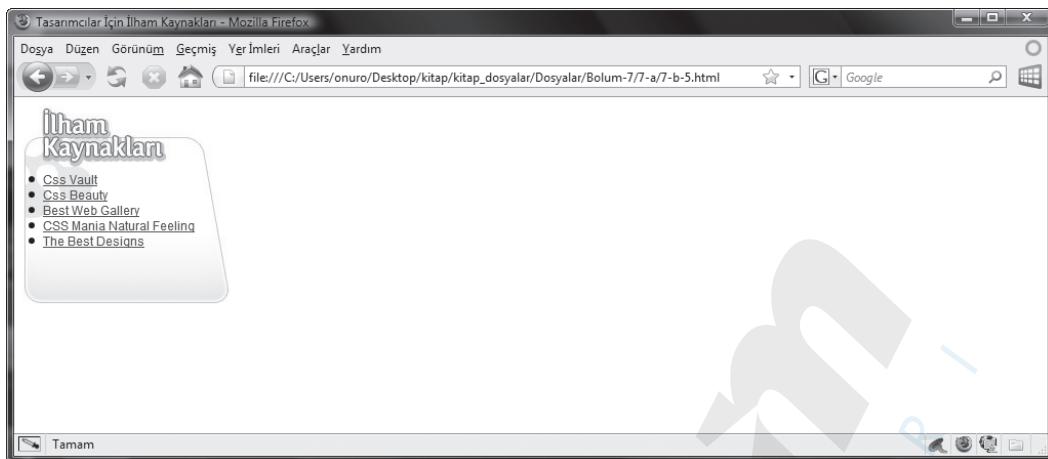


Figür 7-b-4

Arkaplanı atadık, ancak görünen o ki ul elementi üzerinde uygulamamız gereken birkaç şey daha var. Bunlardan en önemlisi ul iç boşuklarını tekrar ayarlayarak liste öğelerini grafiğe uygun bir hizada yerleştirmek. Sonra da ul içi metin özelliklerini ayarlayıp link renklerini daha sonra açacağımız a selektörü içinde biçimleyeceğiz.

Ul metin özelliklerini ve boşluk oranlarını verelim. Bu boşluk oranlarını verirken, sahip olduğu genişlik ve yükseklik değerlerine etki edeceğini unutmuyoruz ve onları da uyumlu hale getiriyoruz.

```
ul {
margin:0; /*dış boşluğu sıfırla*/
padding:65px 0 0 20px; /* üstten 65, soldan 20 piksel iç boşluk */
*width: 182px; /*soldan gelen 20 piksel iç boşluk değerini 202
piksel genişliğinden çıkar (sonuç itibariyle ul genişliği fiziki
olarak yine de 202dir)*/
height: 130px; /* üstten gelen 60 piksel iç boşluk değerini 195
piksel genişliğinden çıkar (sonuç itibariyle ul yüksekliği fiziki
olarak yine de 195tir)*/
font: 9pt Arial;
background: url(liste-arkaplan.jpg) no-repeat;
}
```



Figür 7-b-5

Artık yavaş yavaş listemizin görünümü şekillenmeye başladı. Bu arada fark ettiyseniz ul özelliğini ilk defa biçimlerken sıfırladığımız iç boşluk oranı nedeniyle ekranın görüntülenmeyen kısmında kalan liste öğe imleçleri tekrar görünür hale geldi.

Dolayısıyla liste öğelerindeki bu imleçleri yok edeceğiz. Bu aşamaya geçmeden önce liste öğe imleçlerini biçimlemeye yarayan CSS özelliğinden, **list-style-type** özelliğinden bahsetmek istiyorum:

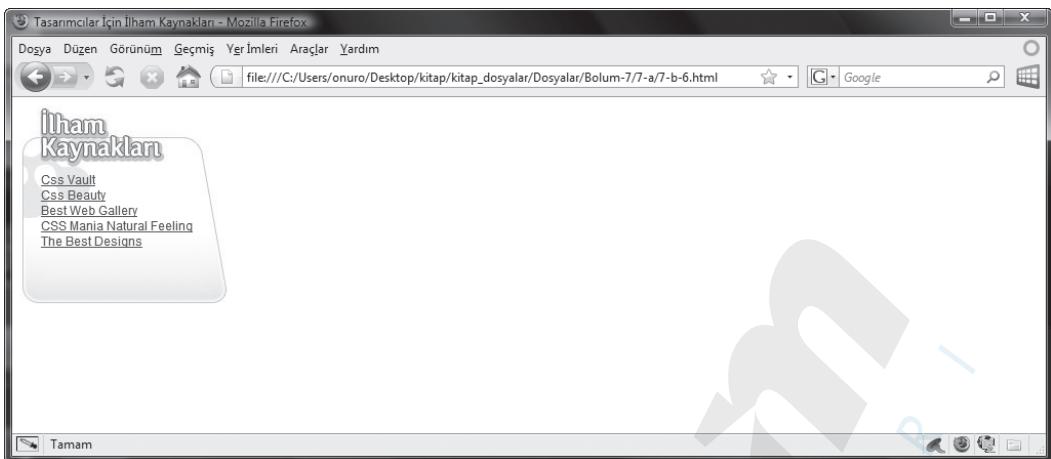
Bu özellik, dokümanımızdaki liste öğelerinin sahip olduğu imleçlerin türünü biçimlememize yaramakla beraber, az sonra ihtiyaç duyacağımız gibi onları kaldırmayı da sağlar.

list-style-type özelliğine aşağıdaki değerler girilebilir:

- **disc**: disk görünümü
- **circle**: çember görünümü
- **square**: baklava görünümü
- **decimal**: sayısal görünüm
- **lower-roman**: küçük roma rakamlarıyla görünüm
- **upper-roman**: büyük roma rakamlarıyla görünüm
- **lower-alpha**: küçük harflerle alfabetik görünüm
- **upper-alpha**: büyük harflerle alfabetik görünüm
- **none**: liste imlerini kaldırır, göstermez

Biz liste öğelerinin imleçlerini kaldırmak için bu değerlerden **none**'a ihtiyaç duyacağız. Şimdi dierseniz li etiketi için bu uygulamayı yapalım:

```
li {
    list-style-type:none;
}
```



Figür 7-b-6

Sıra link özelliklerini biçimlemeye geldi. Link biçimleme için öncelikle a selektörü açıktan sonra satır yüksekliği (line-height) atayarak link satırları arasına biraz mesafe koyacağiz.

Tahmin edeceğiniz üzere line-height öğelerin içindeki satırların boşluk oranını belirlememize yarayan bir özelliktir. Punto (pt), piksel (px) ve değişken boyut (em) değerlerini alabilir.

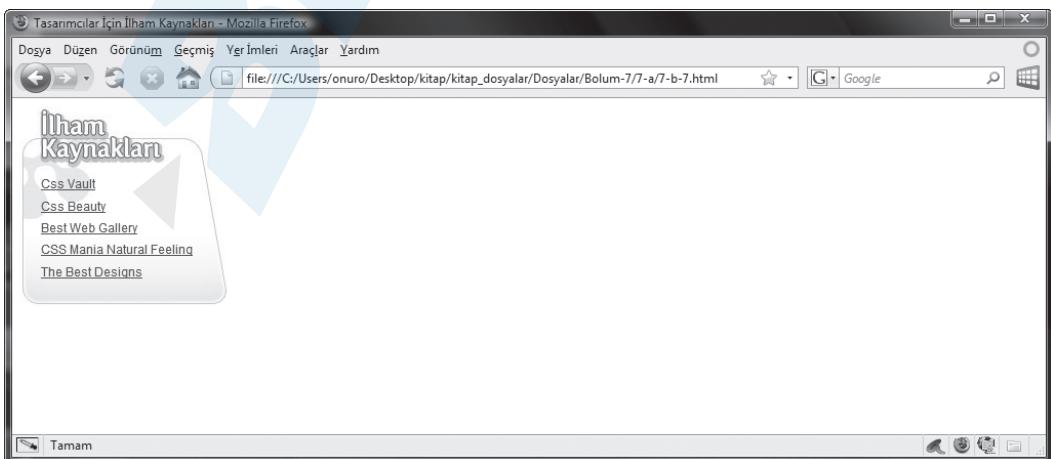
Burada bir diğer konu ise etiketinin yapısıyla ilgili. Link oluşturmamıza yarayan <a> etiketi, her ne kadar yapısı itibariyle kutu seviyesi (**block level**) gibi görünse de, metinler içinde yer alan bir etiket olduğu için fiziki yapısı satır içidir (inline). Bu da linklerin normal şartlarda yalnızca mouse ile link metninin üzerine geldiğimiz zaman aktif hale geleceği anlamına gelir.

Ziyaretçi, link kapsamına girdiği anda tıklama yapamazsa (bilincaltında bunun gerçekleşmesini bekleyecektir), link metninin üzerine gelerek tıklaması gerekecektir. Kendilerini bu zahmetten kurtarmak için, etiketine CSS yardımıyla display özelliğini kullanarak kutu seviyesi etkisi kazandırabiliriz.

Bu işlemi gerçekleştirmek için etiketinin display özelliğine bu kez block değerini atayacağız. Bu, etiketin CSS yüklü görünümde tam olarak kutu seviyesi özelliğine sahip olmasını sağlayacak, ziyaretçi link kapsama alanına girdiği an tıklayabilecektir.

Şimdi, bu yukarıda bahsettiğim özellikleri CSS dokümanımızda bir a selektörünü açıp uygulayalım:

```
a {  
    line-height:16pt;  
    display:block;  
}
```



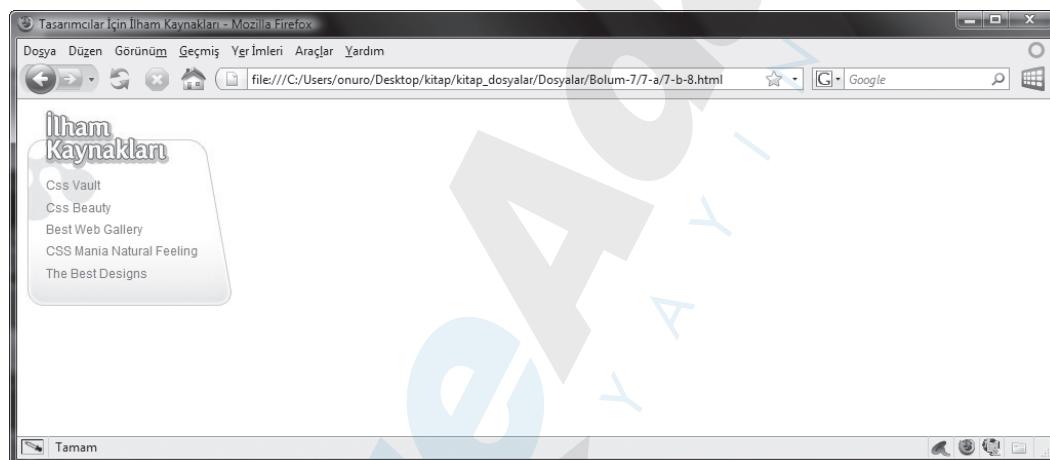
Figür 7-b-7

İsimiz neredeyse bitti.

Son olarak, linklerin renkleri ve fare üzerlerine geldiklerinde alacakları değişikliği uygulayalım. Ben bu örnek için link rengini grafiğin mavi tonlarına uygun bir renk verirken, fare üzerlerine geldiklerinde lacivert rengi alacak şekilde uygulayıp linklerin sahip olduğu alt çizгиyi kaldırıyorum:

```
a {
    line-height:16pt;
    display:block;
    color: # 5c809c;
    text-decoration:none;
}

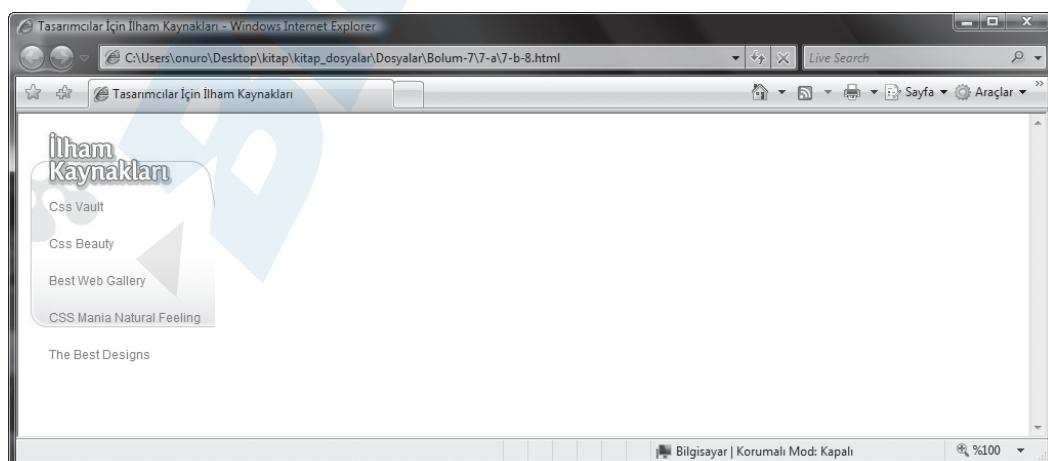
a:hover {
    color:# 23445e;
}
```



Figür 7-b-8

Harika bir sonuç elde ettik değil mi?

Her ne kadar buraya kadar Firefox tarayıcı ile test ettiysek de, Internet Explorer ile baktamızda da fayda var. Aynı dokümanı Internet Explorer ile açalım:



Figür 7-b-9

Görünüşe göre uygulamamız Internet Explorer ile tam bir felaket. Buradaki sıkıntılarla neden olan Internet Explorer handikaplarına bir göz atalım:

1. Arkaplan grafiğinin genişlik ve yüksekliğinden kışkırmız padding değerleri, Internet Explorer'da ul elementinin kesilmesine neden oluyor.
2. Link öğelerine herhangi bir genişlik atamadığımız için Internet Explorer satır yüksekliğini fazladan değer olarak liste öğelerine ekliyor.
3. body elementinin tarayıcı standardı margin oranları Internet Explorer'da farklı yorumlanıyor.

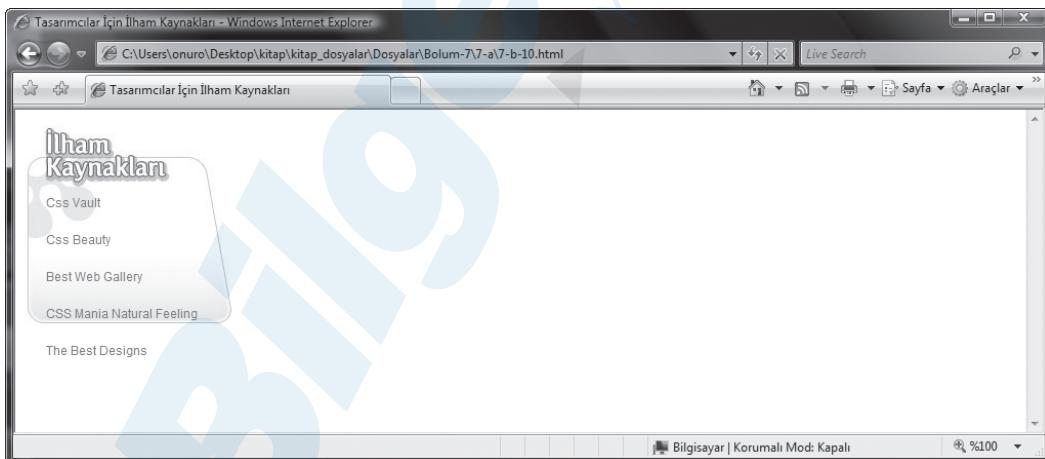
Bu sorunları standart css dosyamızda biçimleme yaparak hem Firefox, hem de Internet Explorer'da ortak görünüm sağlamak oldukça zordur.

İşte burada koşullu tarayıcı stili yüklemesi yöntemine başvuracağız. Öncelikle doküman Internet Explorer ile görüntülenirse kullanacağı kuralları içeren bir koşulu HTML dokümanımıza ekliyorum:

```
<!--[if IE 7]>
<style type="text/css">
/*düzeltmeler buraya gelecek*/
</style>
<![endif]-->
```

Şimdi de öncelikle Internet Explorer için ul genişlik ve yüksekliğini tekrar grafiğin orijinal ebatlarına eşitliyoruz:

```
ul {
width:202px;
height: 195px;
}
```

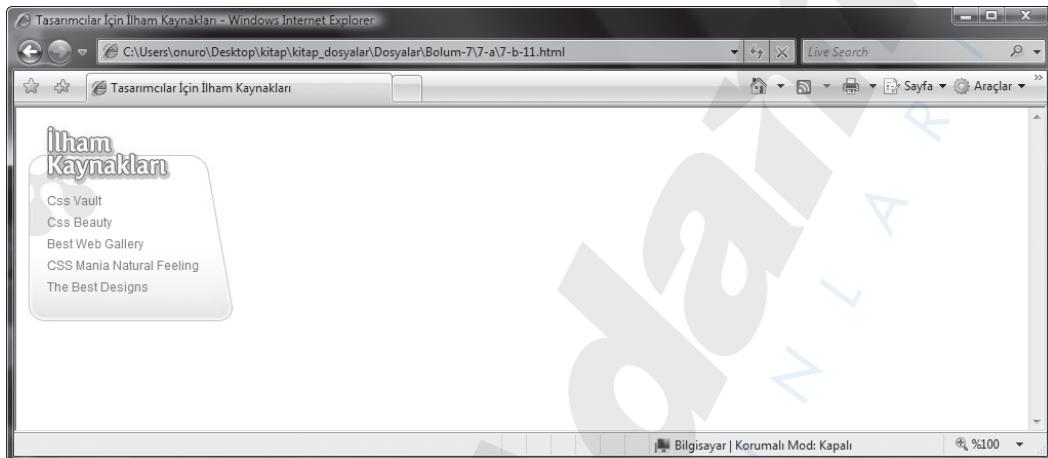


Figür 7-b-10

Liste elementinin genişlik ve yükseklik sıkıntısını çözmüş olduk. Şimdi sıra linklerin genişliğinde. Aslında bunu koşullu stile yerleştirmemize gerek yok. Çünkü orijinal stillendirmemizde linklerin genişliğine dair herhangi bir özelliğe değer atamamıştık.

Orijinal css dosyamızda a etiketinin özelliklerine genişlikle ilgili bir satır daha ekleyelim ve değer olarak, ul'nin soldan iç boşluk oranı olan 20 piksel ebadını grafik genişliğinden çıkardığımızda gelen oluşan sonucu (**182px**) uygulayalım. Böylece linkler ul alanı boyunca kapsamaya devam edecek:

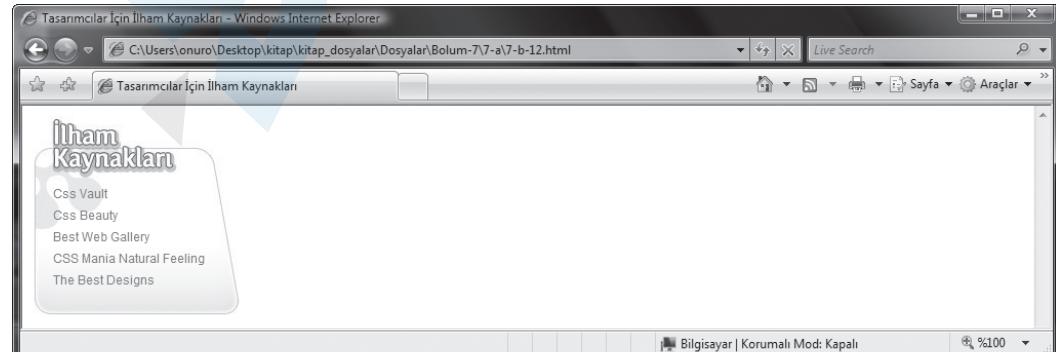
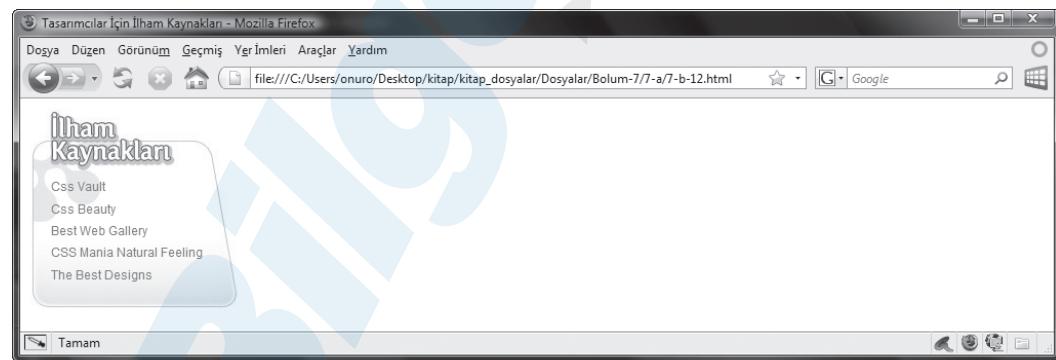
```
a {
    line-height:16pt;
    display:block;
    color: #5c809c;
    text-decoration:none;
    width:182px;
}
```



Figür 7-b-11

Son olarak, gene orijinal CSS dosyamızda body elementi için bir selektör ekleyip margin özelliğine 10px girerek tüm tarayıcılar dört taraftan sabit ve ortak boşluk verelim:

```
body {
    margin:10px;
}
```



Figür 7-b-12

Oldukça ufak piksel kaymaları haricinde neredeyse birebir sonuç elde etmiş olduk.

Girdiğimiz ek biçimlemeleri elbette koşullu stil yüklemesi içinde de kullanabiliyoruz. Ancak CSS profesyonellerinin uyduğu bir kural vardır:

“CSS kullanan profesyonel tasarımcı, Internet Explorer ile olan sıkıntılarını koşullu stillendirme yöntemine gerek kalmadan çözer. Koşullu stillendirme, en son başvurulacak şartır.”

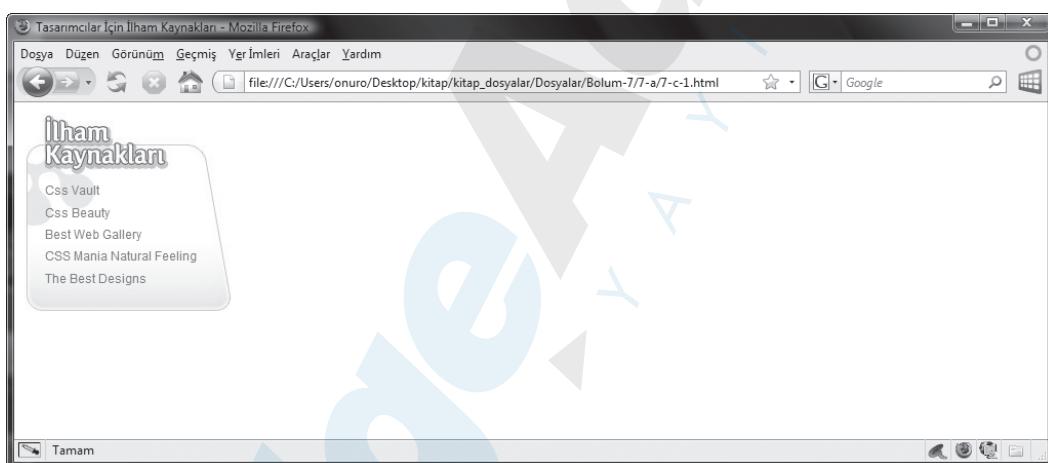
Elbette bu biraz işin esprisi. Ancak Internet Explorer için uyguladığımız ek biçimlemeler dokümanımıza fazladan yük getireceği için, bu sözün doğruluk payı var.

7.c Liste İmleçleri Yerine Grafikler Kullanmak

Görsel bir menü oluştururken her zaman liste öğe imleçlerinin standart görünümünü tamamen kaldırma gereklidir. Bazen, standart imleç görünümü ile birlikte onun yerini de alacak bir grafik kullanmak da o listenin görünümünü oldukça estetik hale getirebilir.

Dolayısıyla bu kez yapacağımız uygulamada, bir önceki örneğimize ek olarak link listele öğelerinin imleçlerini tekrar görünürlük onları bir grafik ile değiştirerek dikey menümüze daha da şık bir görünüm katacağız.

Bir önceki dosyamızda öncelikle bir göz atıp, neleri biçimlememiz gerekiğine bir kez daha bakalım:



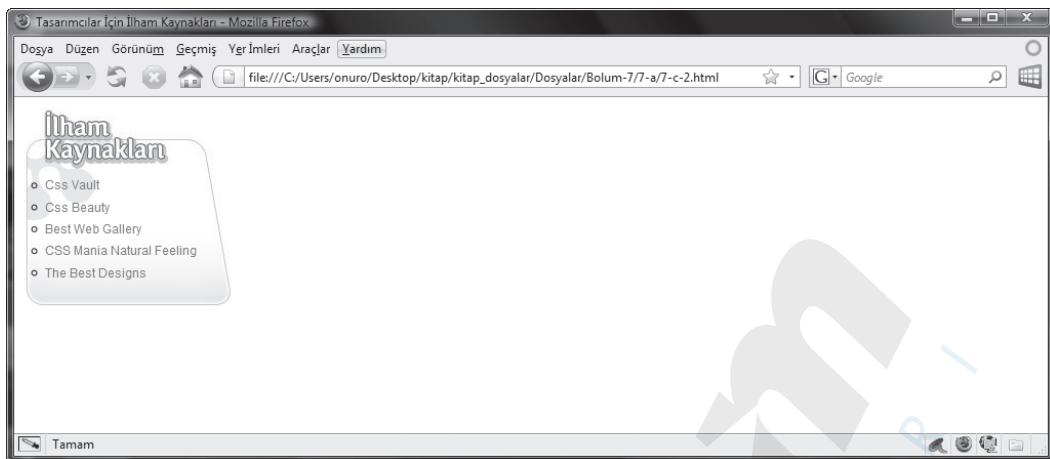
Figür 7-c-1

Yapmamız gerekenler sırasıyla;

- Liste öğelerinin imleçlerini geri kazandırmak
- Öğelerin imleç görüntüsü yerine hazırladığımız grafiği göstermek

Şimdi dilseniz dokümanınızın CSS içerisindeki liste öğelerinin imleç görünümlerini geri kazanmak için `li` etiketimizin selektörünü tekrar biçimlendirelim:

```
li {
    list-style-type: circle /*çember imleç görünümü*/
}
```

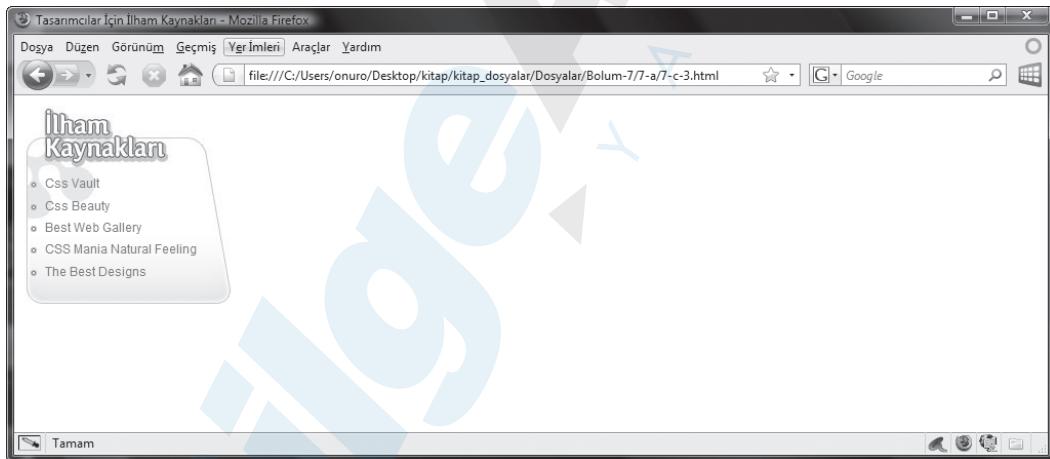


Figür 7-c-2

İmleçlerimiz geri geldi. Şimdi imleçler yerine hazırlanmış olan "imlec.gif" grafiğini göstermemiz gerekiyor.

Bu işlem için, `list-style-image` özelliğine grafiğimizin yolunu vereceğiz:

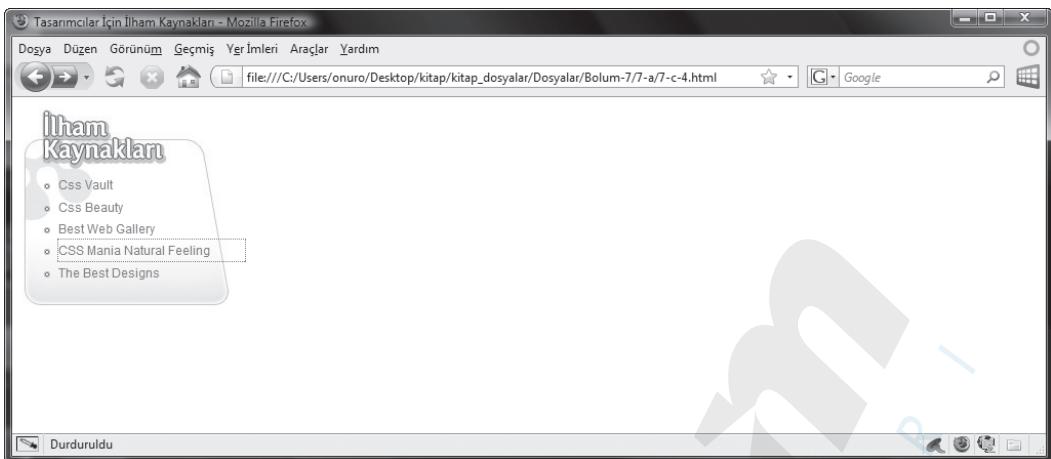
```
li {
    list-style-type: circle; /*çember imleç görünümü*/
    list-style-image: url(imlec.gif);
}
```



Figür 7-c-3

Oldukça sık görünüyor. Ancak sanırım liste öğelerini biraz sola kaydırmamız gerekiyor:

```
li {
    list-style-type: circle; /*çember imleç görünümü*/
    list-style-image: url(imlec.gif);
    margin-left:15px;
}
```



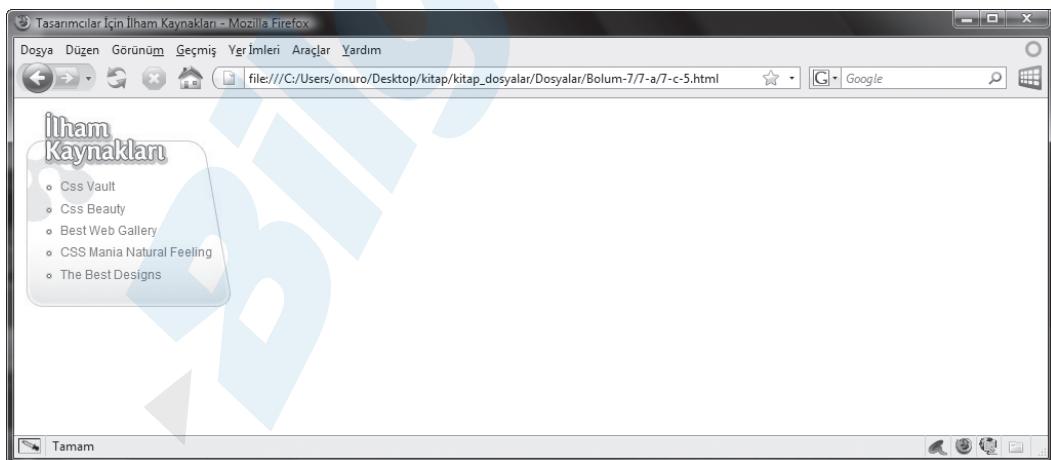
Figür 7-c-4

Görünürde başardık. Ancak dikkat ederseniz, liste öğelerine eklediğimiz fazladan 15 piksellik sol margin, link öğelerinin 182 piksellik genişliğinin ul elementimizin sınırları dışına taşmasına neden oldu.

Eğer ul elementimizin yanında başka bir kutu veya öge olsaydı, bu onun da kaymasına, görünümümüzün muhtemelen bozulmasına yol açacaktı.

Böyle bir riski ortadan kaldırmak için liste öğelerimize eklediğimiz sol margin değerini linklerimizin genişliğinden eksiliyoruz:

```
a {
    line-height:16pt;
    display:block;
    color: #5c809c;
    text-decoration:none;
    width:167px; /* 182(önceki a genişliği) - 15 = 167 */
}
```



Figür 7-c-5

Bu kadar basit.

Bu işlemin bir başka yöntemi de, liste öğelerinin imleçlerini tekrar gizleyip, imleç grafiğini a elementine arkaplan olarak atamak. Bu şekilde muhtemel Internet Explorer sıkıntılarının da önüne geçebiliriz.

7.d CSS ile Metin Tabanlı Yatay Navigasyon Oluşturmak

Yatay menüler, dünyadaki ilk Web siteleri var olduğundan beri kullanılan en işlevsel ve ziyaretçilerin en sık kullandığı fiziksel alanlardır.

Neticede yatay menüler tüm sitenin ana başlık ve içeriklerine erişim olanlığı sağlayan kilit noktalardan biridir. Dolayısıyla bu, site navigasyonu ne kadar problemliyse ziyaretçi o kadar sıkıntı çekecek, ne kadar başarılı ve estetikse ziyaretçi o kadar eğlenceli olacak demektir.

Örnek bir CSS yatay navigasyon uygulamasına göz atalım:



Görüntü 7-d-1

Mozilla Firefox (<http://www.mozilla-europe.org/tr/>) ana sayfasındaki metin tabanlı yatay navigasyon uygulaması, bahsettiğim başarılı stillendirmeye güzel bir örnek.

Elbette daha gelişmiş uygulamaları gerçekleştirmek mümkün. Nitekim onları da ilerleyen örneklerimizde uygulayacağız.

Ancak şimdilik, tekrar liste öğelerini kullanarak, fakat yatay bir şekilde sıralanacak bir dolaşım menüsünün CSS yardımıyla oluşturulma yöntemine deşinelim.

Uygulayacağımız teknikler, bir önceki bölümde yaptığımız sihirbazlıkların neredeyse aynısı. Tek farkla:

Yan yana dizilmeleri gerekiyor.

Tahmin edeceğiniz üzere, `float` özelliğine ihtiyacımız olacak (bir üst elementin yüksekliğini kıracağını da aklımızda tutalım).

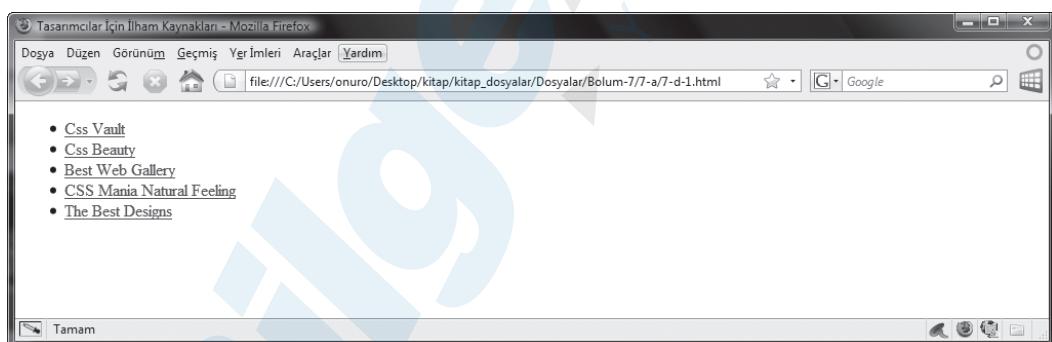
Dilerseniz öncelikle bir önceki dokümanımızı içerisinde `h1` başlığımız bulunmadan tekrar kullanalım:

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns= "http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html" charset="utf-8"/>
<title>Tasarımcılar İçin İlham Kaynakları</title>
<style type="text/css">
/* az sonra buraya css kodlarımızı gireceğiz */
</style>
</head>
<body>
<ul>

<li><a href="http://www.cssvault.com">CSS Vault</a></li>
<li><a href="http://www.cssbeauty.com">CSS Beauty</a></li>
<li><a href="http://www.bestwebgallery.com">Best Web Gallery
</a></li>
<li><a href="http://www.cssmania.com">CSS Mania Natural Feeling
</a></li>
<li><a href="http://www.thebestdesigns.com">The Best Designs
</a></li>
</ul>
</body>
</html>

```



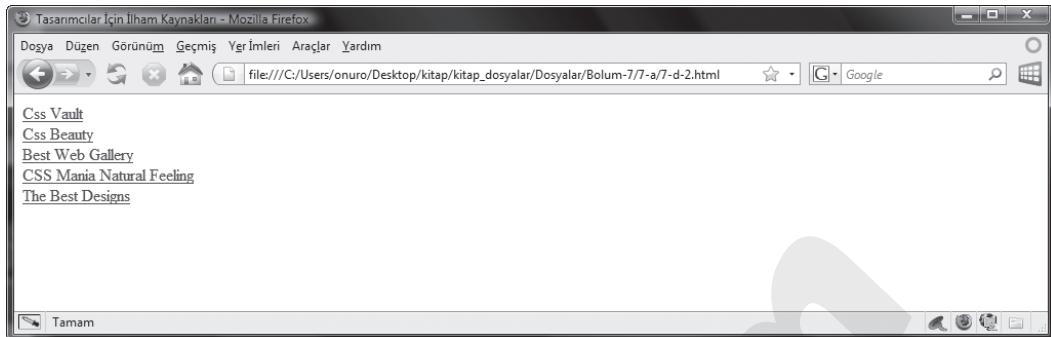
Figür 7-d-1

Öncelikle ul elementinin sahip olduğu tarayıcı standarı boşlukları kaldıralım:

```

ul {
margin:0;
padding:0;
}

```

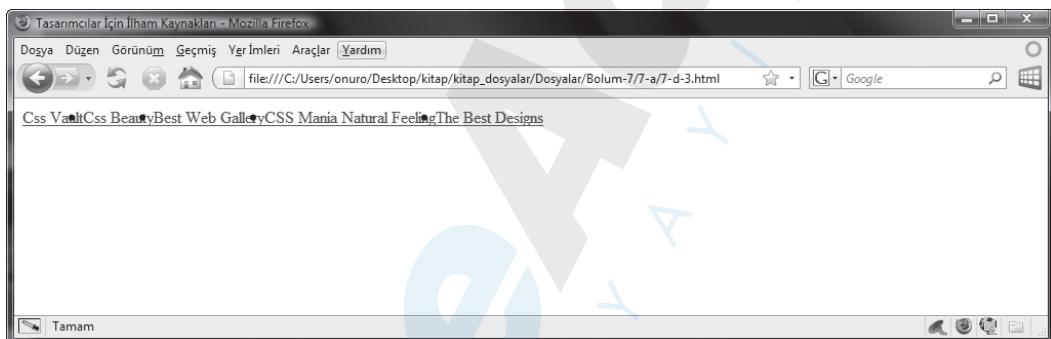


Figür 7-d-2

Ul elementini boşluklarından arındırdık. Bu esnada Firefox tarayıcımızda liste öğe imleçleri de kalkmış görünüyor. Gerçekten öyleler mi, liste öğelerini yan yana yasladığımızda görelim.

Tüm liste öğelerini sola float ettirerek yan yana gelmelerini sağlıyoruz:

```
li {
float:left;
}
```

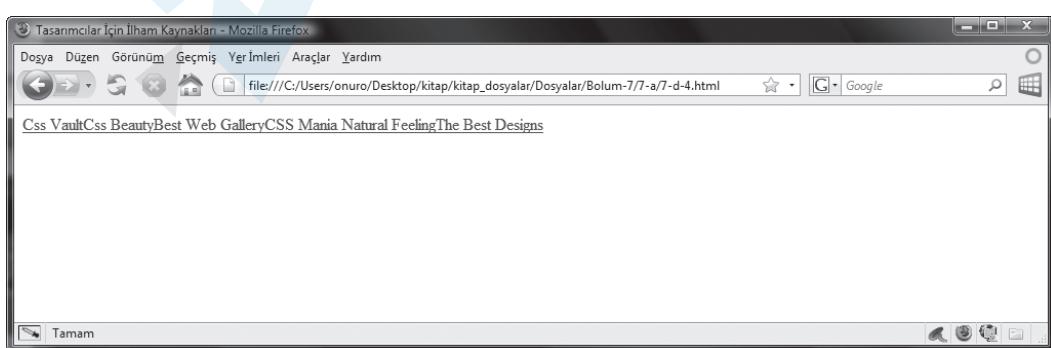


Figür 7-d-3

Tüm linkler böylece yan yana gelmiş oldu. Öte yandan fark ediyoruz ki, liste öğelerinin imleçleri halen oradalar.

Liste öğelerinin imleçlerini kapatıyoruz:

```
li {
float:left;
list-style-type:none;
}
```

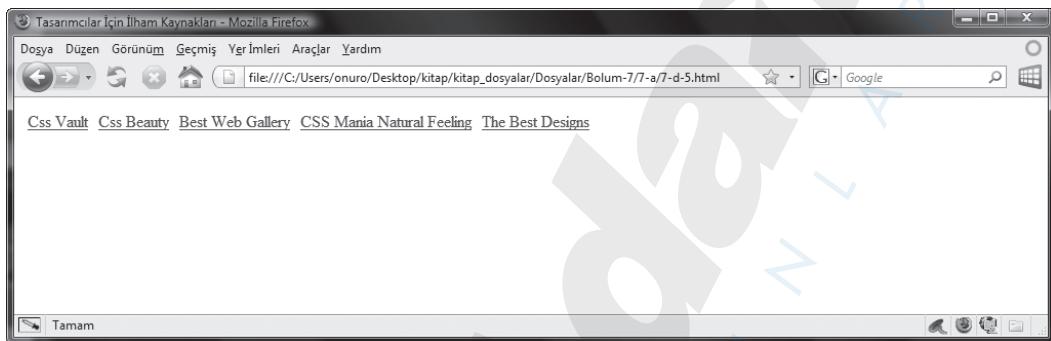


Figür 7-d-4

Şimdi yapmamız gereken öncelikle link elementlerinin (a) tıklama alanlarını `display:block` özelliğiyle link alanı kadar kapsamak, ve elementlere biraz iç boşluk (`padding`) vererek hem okunabilir alanı ferahlandırmak, hem de bunu yaparak aynı zamanda linkler arası boşluk oluşturmaktr.

Link elementlerine kutu seviyesi özelliği kazandırıp 5 piksellik iç boşluk uyguluyoruz:

```
a {  
display: block;  
padding: 5px;  
}
```

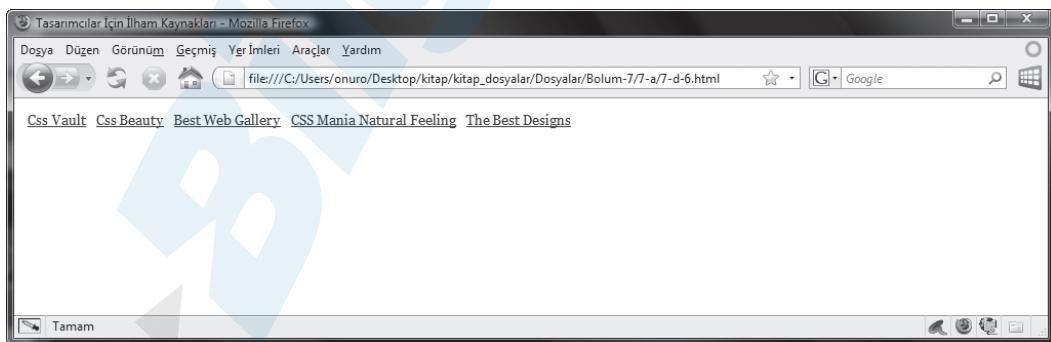


Figür 7-d-5

Yatay navigasyonumuz yavaş yavaş şekil almaya başladı.

Şimdi de, linklerimizin renk ve yazıtını belirleyelim:

```
a {  
display: block;  
padding: 5px;  
font: 9.5pt Georgia;  
color: #000099; /*lacivert */  
}
```

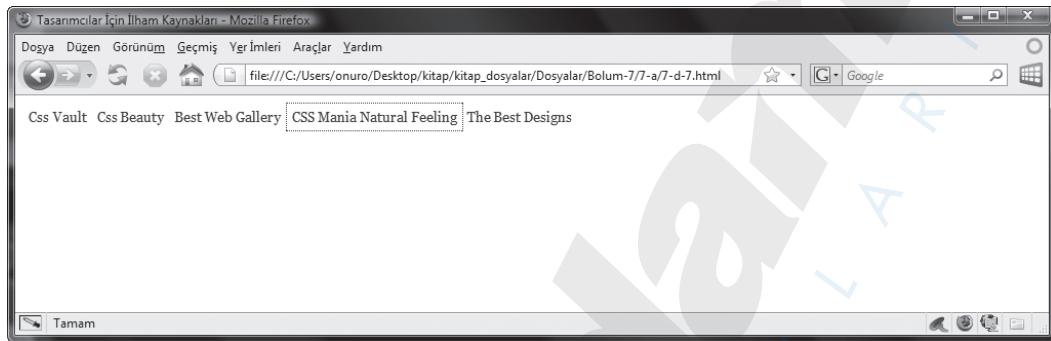


Figür 7-d-6

Menümüz uygulanan rengi ve yazıtipiyle artık daha da estetik görünüyor.

Şimdi linklerimizin altlarında bulunan çizgiyi kaldırıralım:

```
a {
display:block;
padding:5px;
font:9.5pt Georgia;
color:#000099; /*lacivert */
text-decoration:none;
}
```

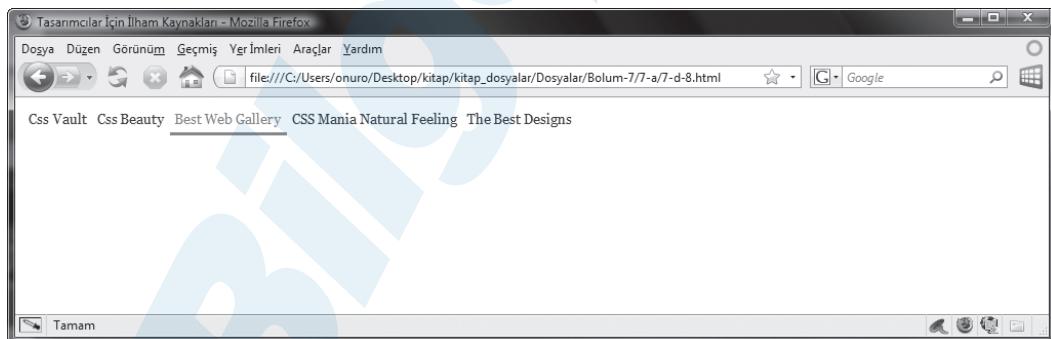


Figür 7-d-7

İşimiz neredeyse bitti gibi.

Dilerseniz son olarak, fare işaretçisi linklerin üzerinde geldiğinde nasıl bir değişiklik gerçekleşeceğini belirtelim:

```
a:hover {
color:#CC6600;
border-bottom:3px solid #CC6600;
}
```



Figür 7-d-8

Böylelikle, oldukça sık, metin tabanlı bir yatay dolaşım menüsü oluşturmuş olduk.

Elbette uygulayabileceğimiz farklı stillendirmelerle, bu tarz menülere binlerce farklı görünüm kazandırmak mümkündür.

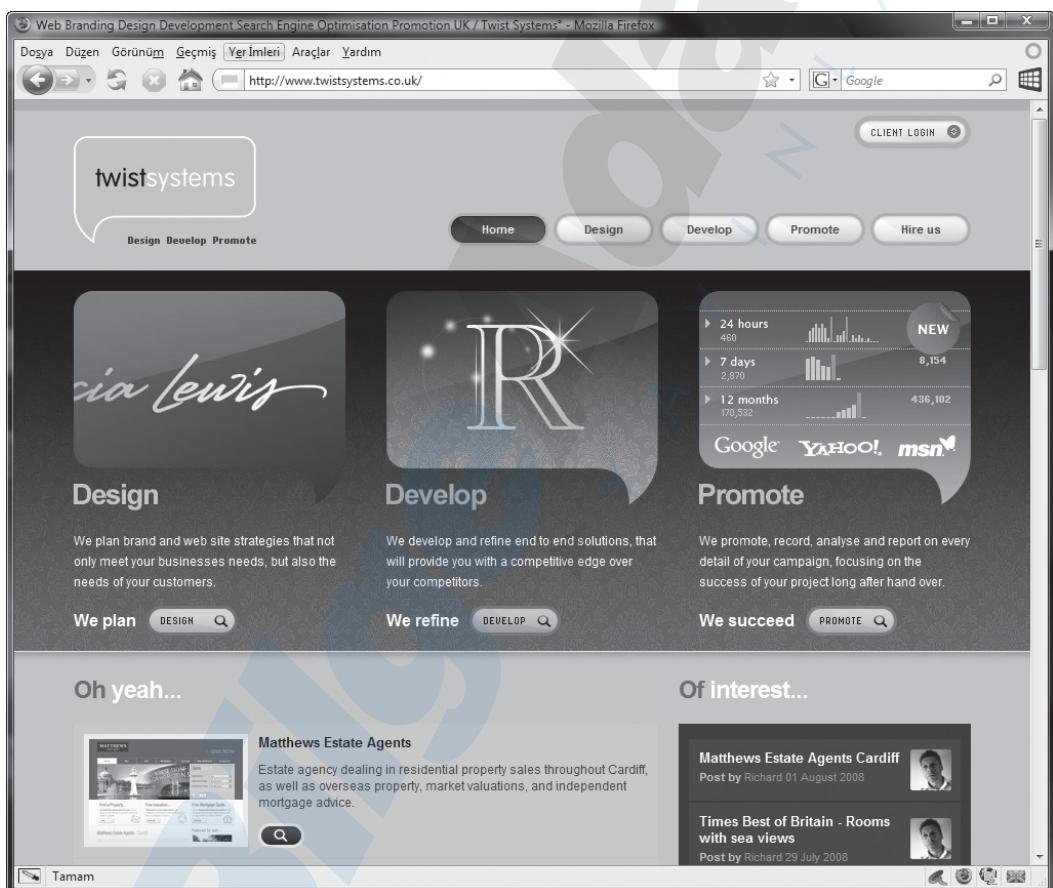
7.e CSS ile Yatay Sekmeli (Tabbed) Grafik Tabanlı Navigasyon Oluşturmak

Bir önceki bölümde de belirttiğim gibi, yatay menüler Web sitelerinin vazgeçilmez parçalarından birini oluşturuyorlar.

CSS'in getirdiği yenilikler, hazırladığımız görsel uygulamaların giderek daha da yaratıcı olabilemesi için müthiş olanaklar sunuyor.

Bir önceki bölümde uyguladığımız metin tabanlı yatay menünün estetiğini, tamamen grafik tabanlı görsel bir navigasyon çalışmasına çevirerek ziyaretçilerimize en ileri seviyede albeniye sahip bir hale getirebiliriz.

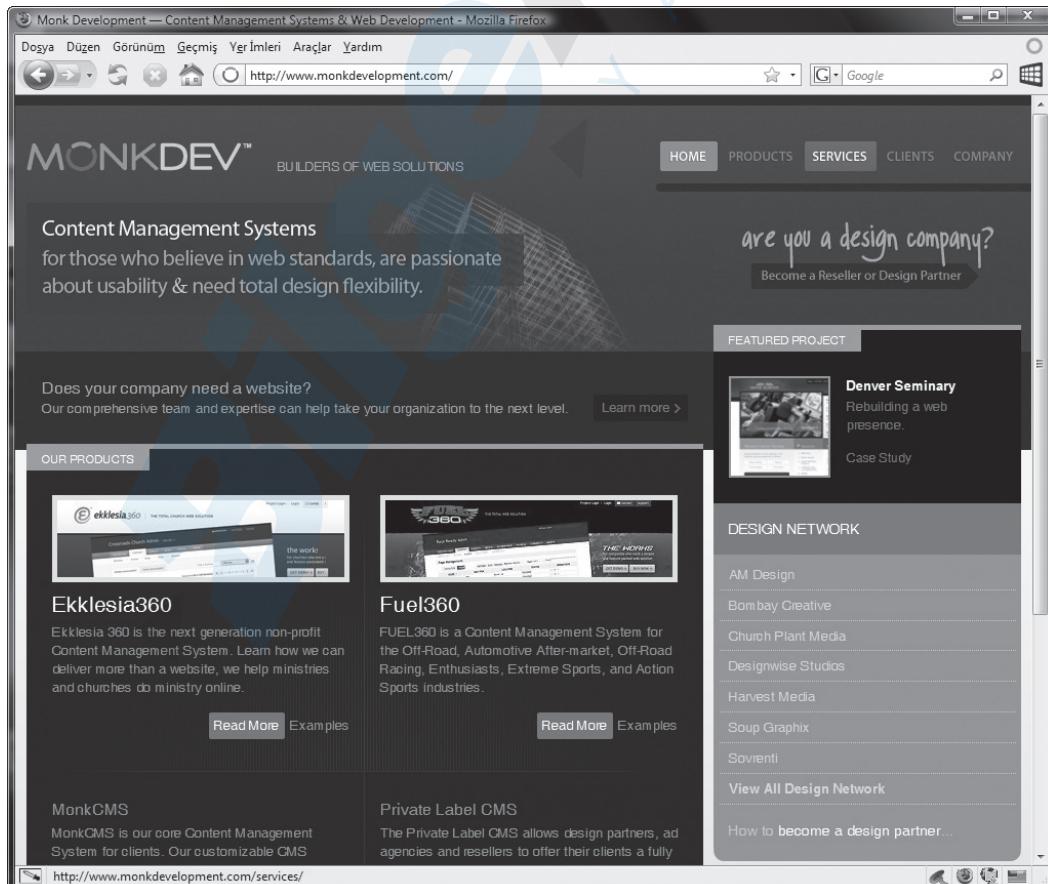
Şimdi size göstereceğim 3 sitenin yatay menüleri, grafik tabanlı yatay menülere ve uygulama alanlarına oldukça güzel birer emsal teşkil ediyorlar.



Görüntü 7-e-1: Twist Systems (www.twistsystems.co.uk)



Görüntü 7-e-2: Iceberg (www.geticeberg.com)



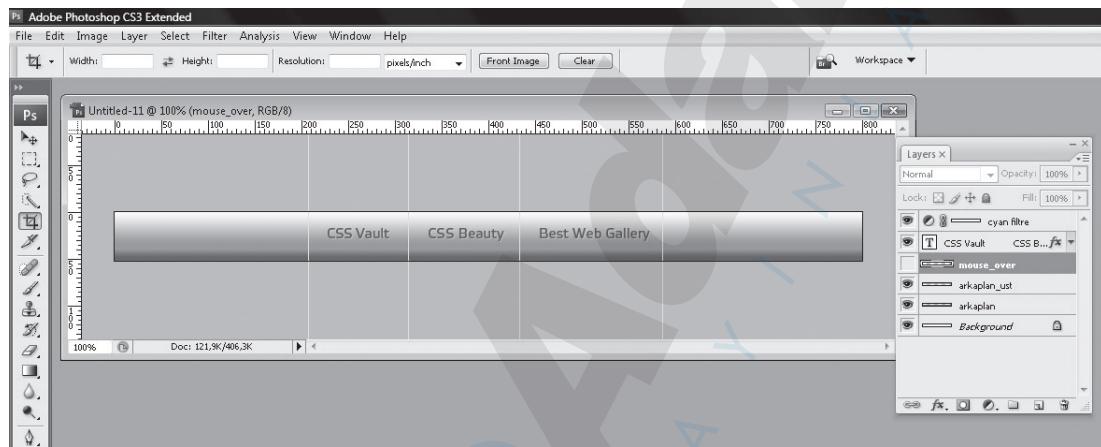
Görüntü 7-e-3: Monk Development (www.monkdevelopment.com)

Üçünde de fark edileceği üzere, menü link içerikleri metin olarak hazırlanmışsa da, hiçbir sistem yazıtılı değil. Bu, onların her parçasının Photoshop veya benzeri bir görsel editörde hazırlanıp, metin tabanlı yatay menü oluşturulduğundan sonra metinler kapatılarak yerlerine bu grafiklerin oturtularak hayatı geçirildiği anlamına gelir.

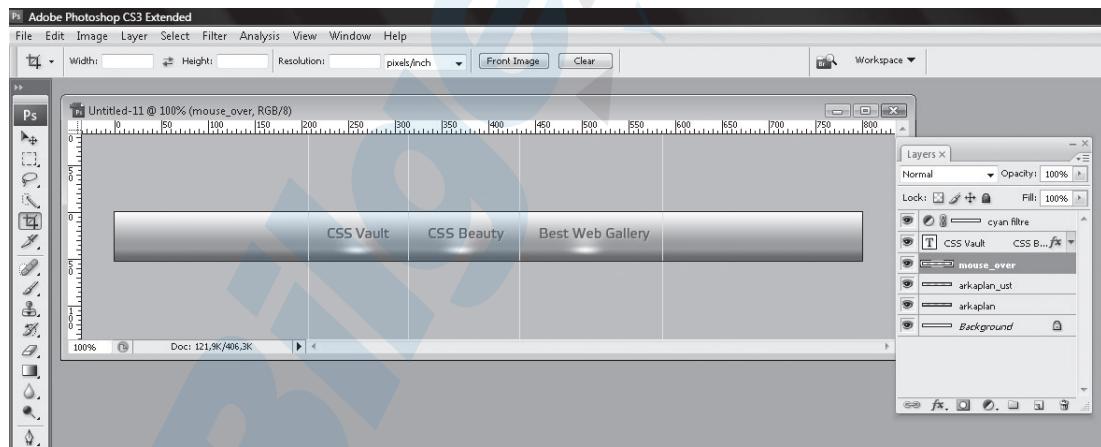
Nitekim biz de aynı yöntemi uyguluyor olacağız.

“Görsel Yerdeğşim / Image Replacement” teknğini evvelki uygulamalarımızdan hatırlayacağınız. Grafik tabanlı yatay navigasyon oluşturmak için, metin tabanlı yatay menü stillendirme tekniklerinin çoğunu aynen uygularken, ek olarak metinler yerine grafik arkaplan atamaları yaparak “görsel yerdeğşim” yöntemiyle yatay menümüzü son derece sık bir görünümeye kavuşturacağız.

Bununla ilgili örneğimizi uygulamak üzere, ben bir önceki listemizden 3 link başlığının CSS ile işimiz bittiğinde nasıl görünmesi gerektiğini, mouseover (hover) durumları ile birlikte Photoshop'ta parçalanmaya hazır grafikler olarak hazırladım:



Görüntü 7-e-4: Normal Görünüm



Görüntü 7-e-5: Linklerin Hover Görünümü

Bu yapının nasıl hayatı geçeceğine yönelik mantık yüretecek olursak; en basit ifadeyle önceliğe Photoshop'ta hazırlanan bu grafikleri rehber hatlarından itibaren keserek kaydettikten sonra, linklerin her birine özel birer arkaplan grafiği olacak şekilde yerleştirerek amacımızı gerçekleştirebileceğimizi söyleyebiliriz.

Ancak, bu çözümün hover kısmında, yani ziyaretçinin fare işaretçisini linklerden birine götürmesi esnasında, “**HILD – Hover Image Load Delay (Mouse Over Grafiği Yükleme Beklemesi)**” ile karşı karşıya kalabiliriz.

HILD, tablolu tasarım zamanından kalan mouseover efektlerinin küçük bir kusurudur. CSS'de de geleneksel bir yükleme/gösterme tekniği kullanıldığından günümüzde de halen bu sıkıntiya sahip örnekleri çokça görebiliriz. HILD şöyle gerçekleşir:

1. Mouseover grafikleri, görsel editörde linkin önce normal hali, sonra da fare işaretçisi üzerine geldiği anki hali ayrı ayrı grafikler olarak kaydedilir. (**Örn:** button_anasayfa_koyu.jpg / button_anasayfa_acik.jpg)
2. Bu grafikler elementin normal (a) ve hover (a:hover) selektörlerine ayrı ayrı grafikler olarak atandığı için, CSS yüklemesi sonrası ziyaretçi menüyü gezerken linkin ilk hali grafik olarak sorunsuz yüklenir.

Ancak ziyaretçi fare işaretçisini o linkin üzerine getirene kadar ikinci hal (hover grafiği) yüklenmeyecektir. Ziyaretçinin hover grafiğini görebilmesi için grafik dosyası yüklenene kadar fare işaretçisini linkin üzerinde tutmak zorundadır.

Bu nedenle özellikle grafikler belli bir yükleme boyutunun üzerindeyse, ziyaretçinin hover grafiklerini asla görmeden gezdiği durumlar dahi gerçekleştirilebilir.

Bu da, böyle durumlarda tasarımcının hover grafiği için yok yere fazladan çaba göstermiş olmasına anlamına gelir.

Bu durumu kolayca çözmenin yolu, başka bir hover teknigi kullanmak.

Sliding Doors / Kaydırılan Arkaplanlar Tekniğiyle Grafik Tabanlı Yatay Navigasyon

Ziyaretçi, HILD etkisine maruz bırakmadan grafik tabanlı hover efektlerine sahip bir yatay navigasyon oluşturmanın en başarılı yöntemi, link grafiklerinin normal ve hover durumlarını ayrı grafikler olarak değil, tek dosya olarak kaydetmektir.

Keza benim hazırlamış olduğum çalışmada da, linklerin bulunduğu kutu ayrı bir arkaplan olarak, link grafiklerinin normal ve hover durumları da tek grafik dosyası olarak kaydedilmiş durumda olacaklar.

Bu grafikleri CSS uygulamaya hazırlamanın tekniği, görsel editör ile rehber hatlarından itibaren kesilen grafiğin iki katı genişliğinde bir boş canvas alanına yerleştirilerek yanına grafiğin hover durumunun konması ve kaydedilmesinden ibarettir:



Görüntü 7-e-6: btn_css_vault.gif

Link grafik genişliği: 97px

Hover efekti ile birlikte: $97 \times 2 = 241\text{px}$;



Görüntü 7-e-7: btn_css_beauty.gif

Link grafik genişliği: 119px

Hover efekti ile birlikte: $119 \times 2 = 238\text{px}$;



Görüntü 7-e-8:btn_best_web_gallery.gif

Link grafik genişliği: 153px

Hover efekti ile birlikte: 153 x 2 = 306px;

Son olarak da linklerimizin ve liste elementimizin (ul) yerleşeceği kutunun arkaplan grafiği kaydedildi:



Görüntü 7-e-9:menu_arkaplan.gif

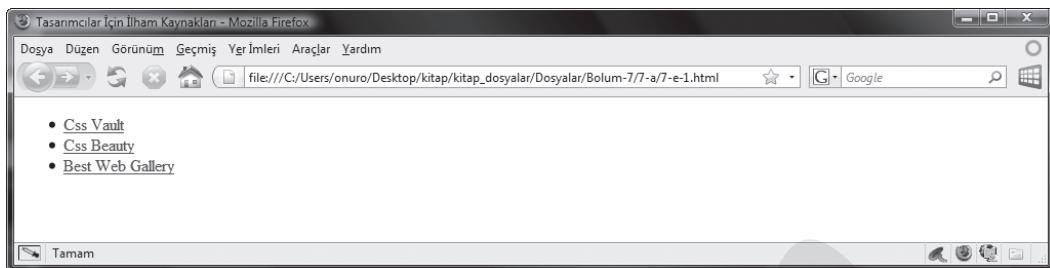
Bütün bu görsellerin hepsinin yüksekliğinin 52 piksele sahip olduğuna dikkatinizi çekmek istiyorum. Bu, listenin içine oturacağı kutunun (div), liste elementinin (ul) ve linklerin (a) hepsinin birden ortak sabit yüksekliğe sahip olacağı, bizim uygulama esnasında yükseklikten yana sıkıntı yaşamayacağımız anlamına gelir.

Şimdi, dilerseniz stillendirmeye geçmeden önce HTML dokümanımızı hazırlayalım:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns= "http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html" charset="utf-8"/>
<title>Tasarımcılar İçin İlham Kaynakları</title>
<style type="text/css">
</style>

</head>
<body>
<div id="menu">
<ul>
<li><a href="http://www.cssvault.com">CSS Vault</a></li>
<li><a href="http://www.cssbeauty.com">CSS Beauty</a></li>
<li><a href="http://www.bestwebgallery.com">Best Web Gallery
</a></li>
</ul>

</div>
</body>
</html>
```



Figür 7-e-1

Şu an itibariyle, stilsiz olarak menümüzün nasıl görüntüleneceğini görüyoruz.

Yapmamız gerekenler sırasıyla şu şekilde:

1. Menü arkaplanımız, tüm doküman genişliğimiz boyunca uzayacağı için, arkaplan grafiğimizi menu div kutumuzun arkaplanı olarak belirleyip bu kutunun genişliğine %100 değer vereceğiz.
2. Liste öğelerini (li) yan yana hizalandıktan ve biçimledikten sonra linklerin (a) kendilerine özel grafik arkaplan tabanlı genişliklerini uygulayıp stillendirmesini yapacağız.
3. Yapılan tasarıma göre liste elementi, yani linklerin bulunduğu liste bloğu kutu içinde ortalanmış halde. Bu durumda liste elementinin (ul) genişliğini, link grafiklerinin genişlik toplamına eşitledikten sonra, ul elementimizi menu kutumuz içinde ortalamamız gerekiyor.

Bu iş için margin özelliğini kullanacağımızı hatırlatmama gerek yok sanırım.

Şimdi öncelikle “menu” div kutusunun özelliklerini tanımlayarak arkaplan atamasını yapalım. Bunu uygularken, dokümanda menüye ait boşluklar oluşmaması için doküman geneli boşluğunu da sıfırlamamız gerekiyor:

```
body {
margin:0;
padding:0;
}

#menu {
width:100%; /*doküman boyunca uzunluğu devam ettir*/
background:url(images/menu_arkaplan.gif) repeat-x; /* "images" klasöründeki menu-arkaplan.gif grafiğini ata, yatay olarak tekrar ettir*/
height:52px; /*menünün grafik yüksekliği*/
}
```



Figür 7-e-2

Olabilecek muhtemel boşlukları body etiketinin margin ve padding standart değerlerini sıfırlamamıza rağmen görüyoruz ki, özellikle tepeden (**ayrıca liste öğeleri için soldan da**) bir kayma mevcut.

Tahmin edeceğiniz üzere, bu durum ul elementinin sahip olduğu standart margin ve padding değerlerinden kaynaklanıyor. Bunları sıfırlayarak kayma problemini çözelim:

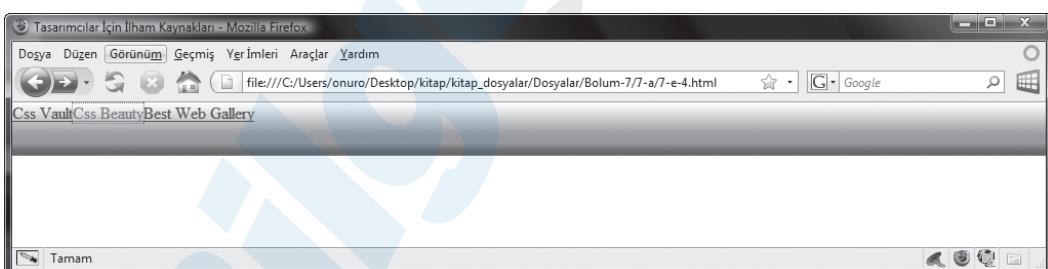
```
ul {
margin:0;
padding:0;
}
```



Figür 7-e-3

Böşluk problemlerinden kurtulduğumuza göre, şimdi liste öğeleri için sola yaslama ve imleçleri kaldırma işlemlerini uygulayabiliriz:

```
ul li {
float:left;
list-style-type:none;
}
```

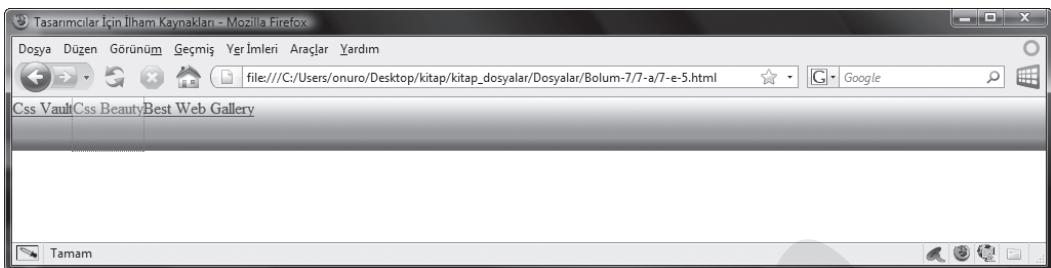


Figür 7-e-4

Şimdi sıra link özelliklerini biçimlemeye geldi.

Öncelikle şunu hatırlayalım; linklerimizin grafik halleri 52 piksel yüksekliğinde. Dolayısıyla bu değeri link özelliklerine de genel atama olarak uygulamamız gerekiyor. Bunun haricinde linklerin metin alanı boyunca değil, kapsama alanları boyunca tıklanabilir olmaları için kutu seviyesine de (block level) çevirmemiz gerekiyor:

```
ul li a {
display:block;
height:52px;
}
```



Figür 7-e-5

Linklere kesin tanımlamayla kutu seviyesi yapısı kazandırıp yüksekliklerini de atadığımıza göre, artık onlara arkaplan özelliklerini sahip oldukları grafik genişlikleriyle atayabiliriz.

Peki, tüm link grafiklerimiz aynı uzunlukta mı? Hayır. Genel link özelliklerine (a) veya liste öğelerine (li) uygulayacağımız genel bir değişiklik problem yaratır mı? Evet.

Dolayısıyla burada şunu fark ediyoruz: Linklerin her biri kendilerine ait özel genişliklere ve arkaplan grafiklerine sahip. Bunu gerçekleştirecek ancak onları bir id selektörü ile kapsayarak yapabiliriz.

HTML üzerinde a etiketleri üzerine doğrudan id ataması yapmak mümkün. Ancak onun yerine, li öğelerine id ataması yaparak biçimlendirmek ileri vadede daha pratik bir çözüm. Buna göre, açacağımız selektörler id adı atanın li elementinin içindeki a elementini biçimleyen selektörler olacak ve biz o selektörlerle gerekli tanımlamaları yapacağız.

Şimdi öncelikle HTML dokümanımızda liste öğelerine gerekli id atamalarını yapalım. Ben isimlenme olarak link adlarını kullandım:

```
...
<li id="nav_css_vault"><a href="http://www.cssvault.com">CSS
Vault</a></li>
<li id="nav_css_beauty"><a href="http://www.cssbeauty.com">CSS
Beauty</a></li>
<li id="nav_best_web_gallery"><a href="http://www.bestwebgallery.
com">Best Web Gallery</a></li>
...

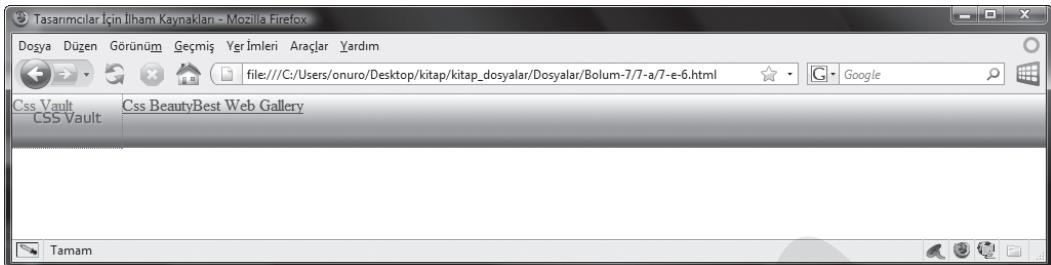
```

Şimdi de öncelikle CSS Vault linkinin grafik arkaplan atamasını, ona özel verilmiş olan "nav_css_vault" selektörünün içindeki link elementine atayalım.

Dikkat ettiyseniz, "nav_css_vault" selektörünü açıp biçimlemeleri doğrudan atayalım demedim. Biçimlenecek linkimiz bu selektör elementinde (li) değil, onun içindeki link elementinde (a).

Uygulayacağımız biçimlemeler ise; öncelikle CSS Vault grafiğinin yarı genişlik oranını ("btn_css_vault.gif" 214 piksel genişliğinde, dolayısıyla 107 piksel onun hover öncesi normal halini görüntüleyecek) link selektörüne atamak, sonra da arkaplan grafiğini tekrar etmeyecek şekilde yerleştirmek:

```
#nav_css_vault a {
width:107px;
background:url(images/btn_css_vault.gif) no-repeat;
}
```



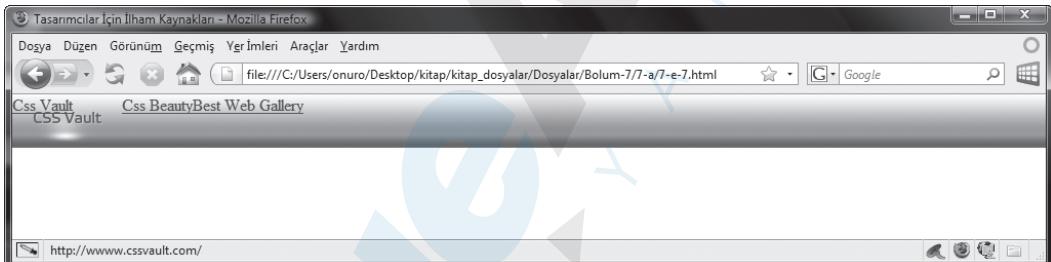
Figür 7-e-6

Yükseklik ve block yapısı genel a selektöründeki biçimlemelerden kalıtsal olarak gelen CSS Vault linki, ona özel atadığımız biçimlemelerle kendisine has görsel yapısına neredeyse kavuşmuş oldu.

Sıra CSS Vault linkinin hover durumunu biçimlemeye geldi. Fare işaretçisi bu linkin üzerine geldiğinde yapacağımız işlem çok basit, **Sliding Doors**: Linkin arkaplanını sağ tarafa pozisyonlandıracak kaydırmak.

Arkaplanı sağa kaydırma işlemini "nav_css_vault" içindeki linkin hover durumu için uygulayalım:

```
#nav_css_vault a:hover {  
background-position:top right;  
}
```



Figür 7-e-7

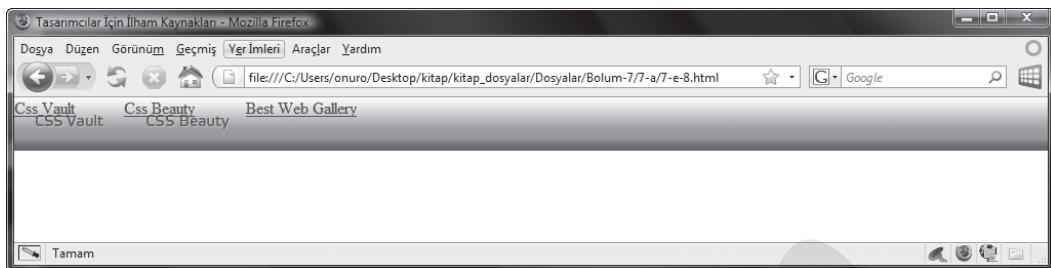
Oldukça etkileyici, değil mi?

Bu linkin normal ve hover durum grafiği tarayıcıda yalnızca bir kez yükleniyor ve ziyaretçi linkin üzerine geldiği anda hover grafik kaydırması ile arkaplan yeni bir görünümeye kavuşuyor.

Metnimiz ise halen linkimiz üzerinde durmaya devam ediyor. Bu metni ve diğerlerini az sonra toptan gizleyeceğiz.

Şimdi ikinci linkimiz olan "CSS Beauty" ögesini kendine ait değerleriyle aynı özelliklere sahip olacak şekilde biçimleyelim (138 piksel grafik genişliği / 2 = 119 piksel genişlik):

```
#nav_css_beauty a {  
width:119px;  
background:url(images/btn_css_beauty.gif) no-repeat;  
}  
#nav_css_beauty a:hover {  
background-position:top right;  
}
```



Figür 7-e-8

CSS Beauty linki için de hem normal durum, hem de hover durumunu kapsayan değişiklikleri uygulamış olduk.

Şimdi de son linkimiz olan “Best Web Gallery” için gerekli stillendirmeyi uygulayalım:

```
#nav_best_web_gallery a {
    width:153px;
    background:url(images/btn_best_web_gallery.gif) no-repeat;
}
#nav_best_web_gallery a:hover {
    background-position:top right;
}
```

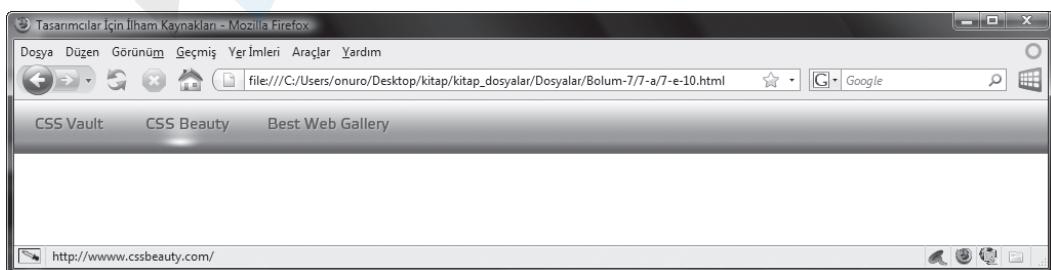


Figür 7-e-9

Menümüz yavaş yavaş şekillenmeye başladı.

Şimdi dilerseniz linkler üzerinde bulunan metinleri CSS yüklü görünümde kaldıralım. Bu işlem için ul içindeki genel a selektörümüze gizleme özelliğini ekleyeceğiz:

```
ul li a {
    display:block;
    height:52px;
    text-indent:-9999px;
}
```



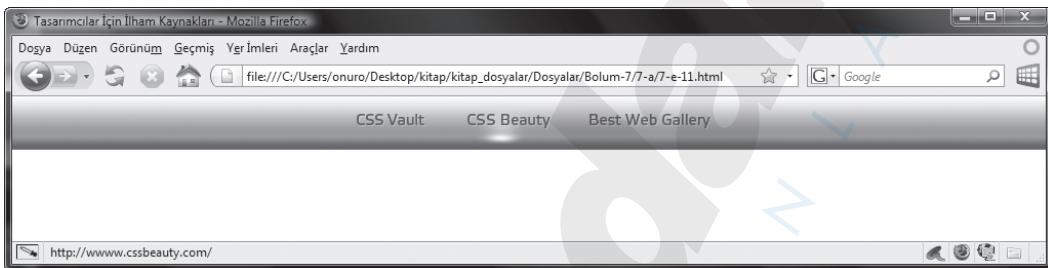
Figür 7-e-10

Buraya kadar fevkalade görünüyor.

Son bir işimiz daha var: Bu link bloğunu (`ul`) ortalamak. Bu işlem için, tüm linklerin yarı genişlik değerlerini toplayıp `ul` elementinin genişliğine atadıktan sonra bu elementi ortalarsak işimiz bitmiş olacak:

CSS Vault (107 piksel) + CSS Beauty (119 piksel) + Best Web Gallery (153 piksel) = 379 piksel

```
ul {
margin:0 auto;
padding:0;
width:379px;
}
```



Figür 7-e-11

İşte bu kadar.

Photoshop ile sekmeli menünün tasarımından başlayan serüvenimiz, menünün birebir olarak **Sliding Doors** teknigi ile CSS kullanılarak hayatı geçmesiyle tamamlandı.

Bu teknik, CSS ile görsel menüler oluşturma amaçlı en popüler yöntemdir. Biz bu örneğimizde yatay menü yapısı için yatay kaydırma uygulayarak gerçekleştirdik. Aynı işlemi dikey kaydırma planlayarak da yapabiliydik.

Sliding doors teknigini sadece menü yapılarında değil, görsel ve hover durumuna sahip olacak her türlü link elementi içerisinde aynı mantıkla uygulayabiliriz.

Liste öğelerini kullanarak, hem erişilebilir hem de son derece şık görsel menüler oluşturma yöntemlerini görmüş olduk. Tekrar belirtmek istiyorum; CSS ile yapabileceğimiz görsel sihirbazlıkların sınırı yoktur.



8

CSS Konumlandırma 2: Relative, Absolute ve Fixed Konumlandırma

8 CSS Konumlandırma 2: Relative, Absolute ve Fixed Konumlandırma

- Elementlerin Normal Akış ve Konum Düzeni
- Bağıl (Relative) Konumlandırma
- Mutlak (Absolute) Konumlandırma
- Hizalanmış İçerik Düzeni için Relative ve Absolute Konumlandırmayı Birlikte Kullanmak
- Sabit (Fixed) Konumlandırma

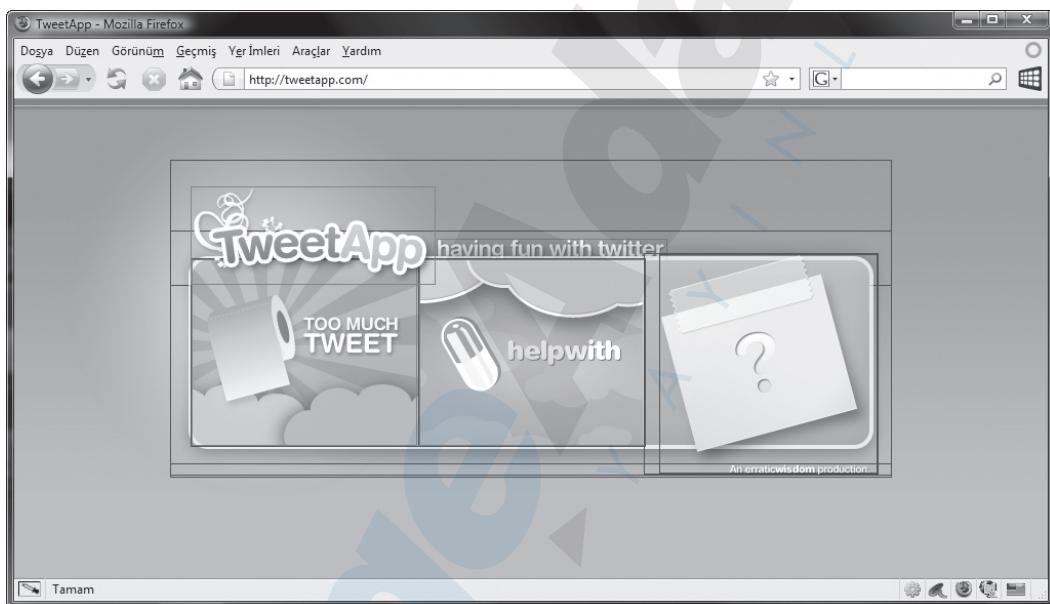
CSS Konumlandırma 2: Relative, Absolute ve Fixed Konumlandırma

Internet'te sıkça rastladığımız drop down (fare ile üzerine gelindiğinde aşağı veya sağa açılan) menüler, sayfaların belli bir noktasına yerleşmiş ve sabit bir şekilde duran elementler tamamen bu başlığı örneklereidir.

Bazı durumlarda bir elementi sola veya sağa yaslamak (float) yeterli gelmeyebilir. O elementi dokümanınızda belirli bir pozisyonaya yerleştirmek isteyebilir veya ne olursa olsun (scroll, büyütme vb.) elementin bulunduğu pozisyonun sabit kalmasını ihtiyaç duyuyor olabiliriz.

İşte bu tip durumlarda, CSS içindeki position özelliğini kullanıyor olacağız.

Daha önceki örneklerde dile getirdiğim Tweetapp sitesinde kullanılmış olan position özellikli elementlere bir göz atalım:

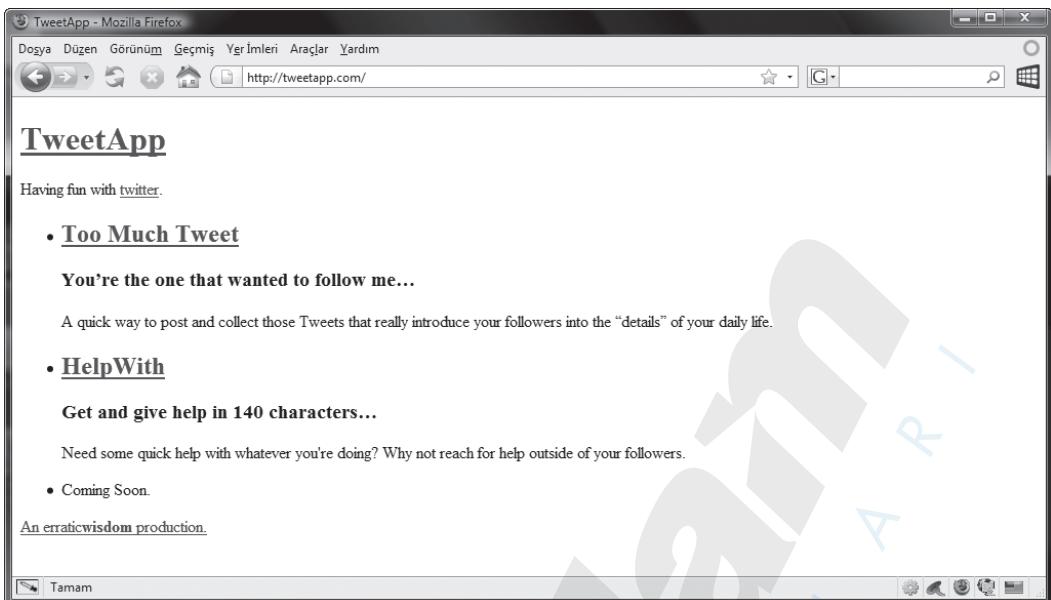


Görüntü 8.1

Çerçevelelerle işaretlenmiş olanlar, position özelliği kullanılarak hizalanmış elementleri temsil ediyor.

Bunlardan kırmızı ile çerçevelenmiş olanlar, ana kutuları içinde mutlak (**absolute**), mavi olanlar ise bağıl (**relative**) konumlandırmaya sahipler. Bu stillendirmeler, o elementlerin float ile yapılabileceğinden daha pratik bir şekilde konumlandırılabilmelerini sağlıyor.

Siteyi stilsiz görüntüleyerek bu görüntüyü sağlayan HTML dokümanının ham haline bakalım:



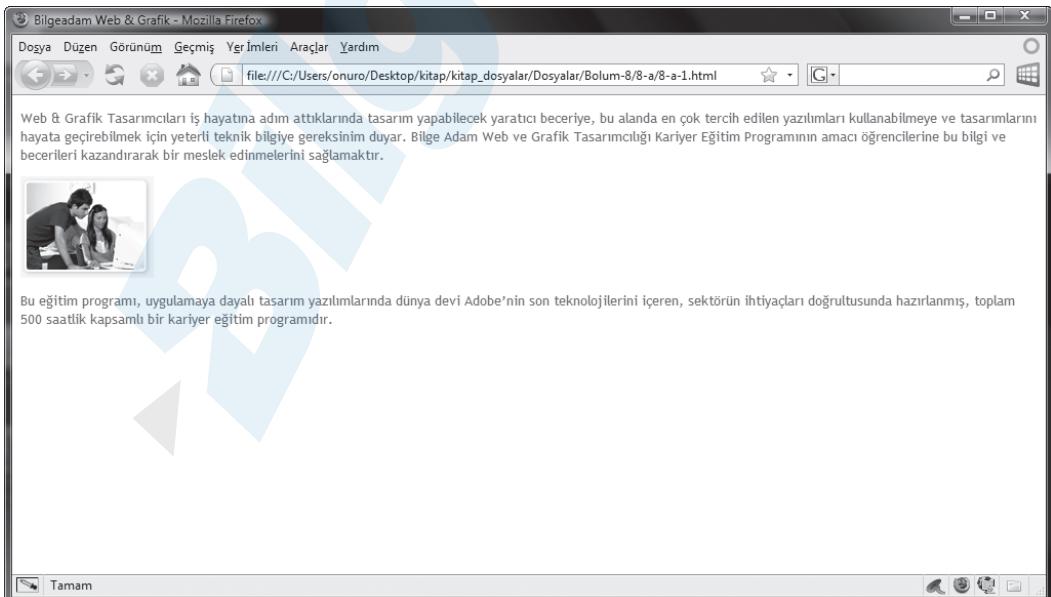
Görüntü 8.2

Elbette position özelliği kullanılarak yapısal konumlandırmaları gerçekleştirmek her zaman gereklili ya da mecburi değildir. Zira uygularken de göreceğimiz üzere, float yöntemine oranla daha hassas davranışımız gereken noktalarla karşılaşacağız.

8.a Elementlerin Normal Akış ve Konum Düzeni

position özelliğiyle ilgili uygulamalarımıza geçmeden önce, bu konuya doğrudan bağlantılı başka bir gerçekten bahsetmek istiyorum.

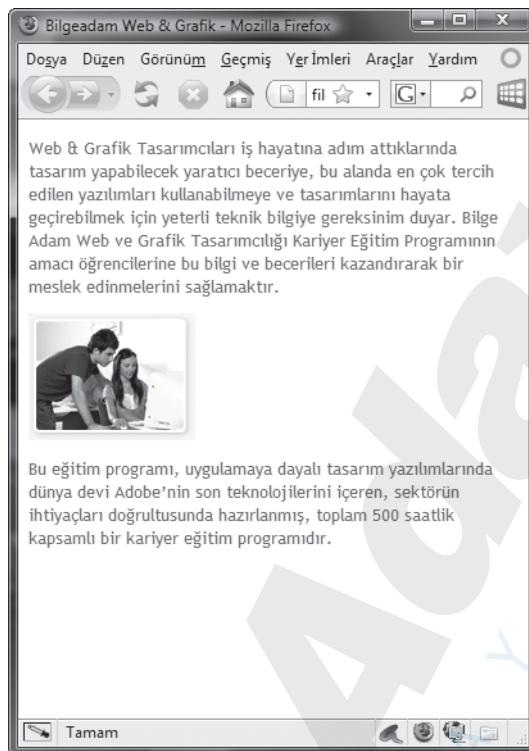
Evvvelce, tarayıcıların dokümanları görüntüülerken özel bir değişiklik yapılmamışsa kendi tarama yöntemlerini gerçekleştirerek bize sunduğundan bahsetmiştim. Bu duruma göre stilsiz bir Web dokümanı, tarayıcılar tarafından belirlenmiş standart bir uzunluk, genişlik ve konum akışına sahiptir. Burada özellikle uzunluk değerinin kutu seviyesi (block level) elementler için %100, konum değerinininse sabit (fixed) olduğunu belirtmek istiyorum:



Görüntü 8-a-1

Oluşturulan bir Web dokümanında bulunan elementler tarayıcının bize sunduğu standart yerleşim ve akışla görüntülenirler.

Şimdi tarayıcı penceresinin ebatlarını değiştirecek olursak, içeriğin bulunduğu elementlerin nasıl tekrar şekillendiğini görelim:



Görsüntü 8-a-2

Göründüğü gibi, içerik genişlik ve yerleşimi, tarayıcı penceresinin ebatlarına göre tekrar şekillenerek ziyaretçiye yeniden biçimlenmiş bir şekilde, konumları değişmeden sunuluyor.

`position` özelliğine yapacağımız değişiklikler, tarayıcıların bu sabit ve standart akış görüntüleme mantığıyla doğrudan ilgilidir.

İhtiyaç duyacağımız noktalarda, CSS yardımıyla tarayıcı ebat ve kaydırma (`scroll`) durumları değişse bile, bazı elementlerimizde sabit yerleşimleri bazılarında tarayıcı standartı konuma bağlı değişken yerleşimler uygulayabileceğiz.

Dolayısıyla `position` özelliğiyle konumlandırma yapmak, diğer konumlandırma yöntemlerine göre biraz daha teknik nitelik taşıyor. Yine de mantığını kavradığınızda sıkıntı çekmeyeceğinizden eminim.

8.b Bağıl (Relative) Konumlandırma

Tarayıcıların normal akışı konusunda bilgi sahibi olduğumuza göre, `position` özelliğine atanacak değerlerin tasarımlarımıza ve dokümanımızdaki konumlandırmalara nasıl etki edeceğine göz atabiliriz.

İlk ele alacağımız `position` değeri, diğer konumlandırma değerlerine oranla en karmaşık niteliğe sahip bağıl (relative) konumlandırma üzerine olacak.

İçgündüşel olarak, bağıl konumlandırmaının, uygulanan elementin diğer elementlerin konumıyla olan ilişkisini baz alan bir değer olduğunu düşünebilirsiniz. Bu doğrudur, ancak bununla da sınırlı değildir.

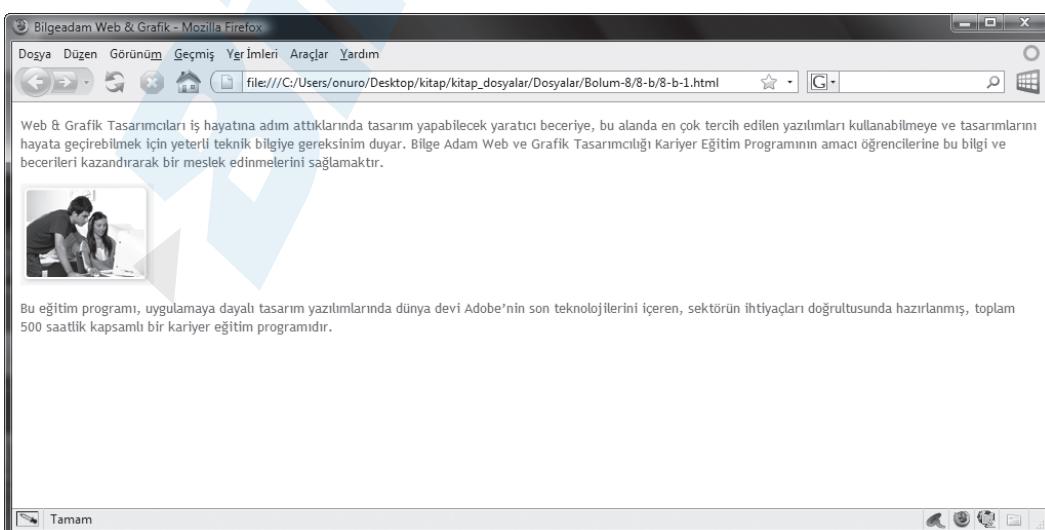
Dilerseniz bağıl konumlandırmaya bir örnekle giriş yapalım. Öncelikle içeriğinde iki paragraf velarında bir görselin () bulunduğu bir HTML dokümanı oluşturuyoruz:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title>Bilgeadam Web & Grafik</title>
<style type="text/css">
body {
font:9.5pt "Trebuchet MS"; /*doküman geneli yazıt tipi özellikleri*/
color:#666666; /*gri metin rengi*/
}
</style>
</head>

<body>
<p>Web & Grafik Tasarımcıları iş hayatına adım attıklarında tasarım yapabilecek yaratıcı beceriye, bu alanda en çok tercih edilen yazılımları kullanılmaya ve tasarımlarını hayatı geçirebilmek için yeterli teknik bilgiye gereksinim duyar. Bilge Adam Web ve Grafik Tasarımcılığı Kariyer Eğitim Programının amacı öğrencilerine bu bilgi ve becerileri kazandırarak bir meslek edinmelerini sağlamaktır.</p>


<p> Bu eğitim programı, uygulamaya dayalı tasarım yazılımlarında dünya devi Adobe'nin son teknolojilerini içeren, sektörün ihtiyaçları doğrultusunda hazırlanmış, toplam 500 saatlik kapsamlı bir kariyer eğitim programıdır. </p>

</body>
</html>
```



Figür 8-b-1

Az sonra, dokümandaki `img` elementinin pozisyonunu bağıl konumlandırmayla (relative) değiştireceğiz.

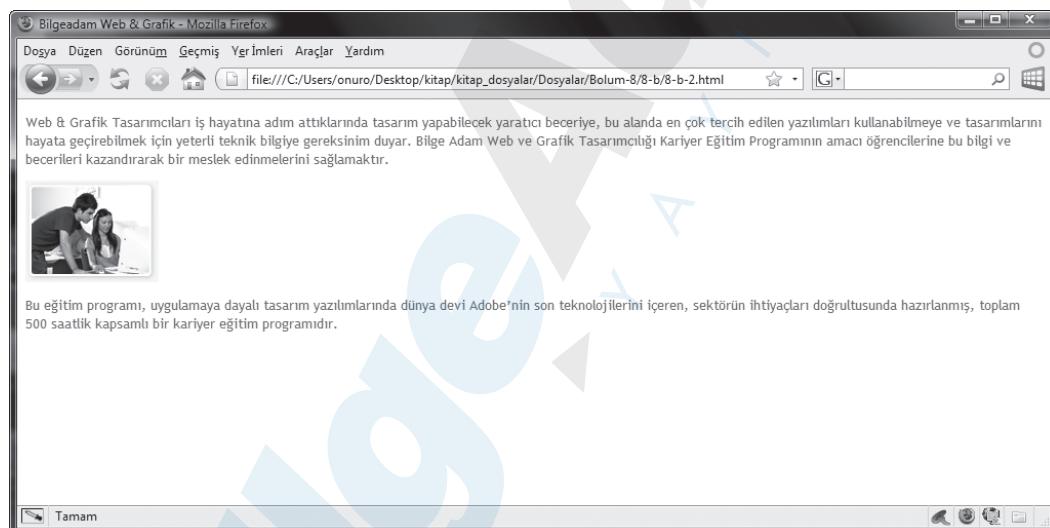
`position` özelliğine girilen değerler çoğunlukla, ofset (göreli konum) özellikleriyle beraber kullanılırlar. Ofset özellikleri margin özelliklerine benzer, yani 4 taraftan mesafeyi belirleme amaçlı kullanılırlar.

CSS içinde kullanılabilecek 4 adet ofset özelliği var, her biri de piksel (px) veya yüzde değerini alabiliyor (%):

- `top`: yukarıdan mesafe
- `left`: soldan mesafe
- `right`: sağdan mesafe
- `bottom`: aşağıdan mesafe

Şimdi dilerseniz dokümanımızdaki görseli (`img`) relative konumlandırma uygulayalım:

```
img {
  position: relative;
}
```



Figür 8-b-2

Göründüğü gibi, uygulanan relative pozisyonu görselimizde henüz herhangi bir konum değişikliği ne neden olmadı. Olabilmesi için, ofset konumları belirlememiz gereklidir.

Biz, ofset konumu olarak sol ve yukarı özelliklerine **150 piksel** değerlerini verelim:

```
img {
  position: relative;
  top: 150px;
  left: 150px;
}
```

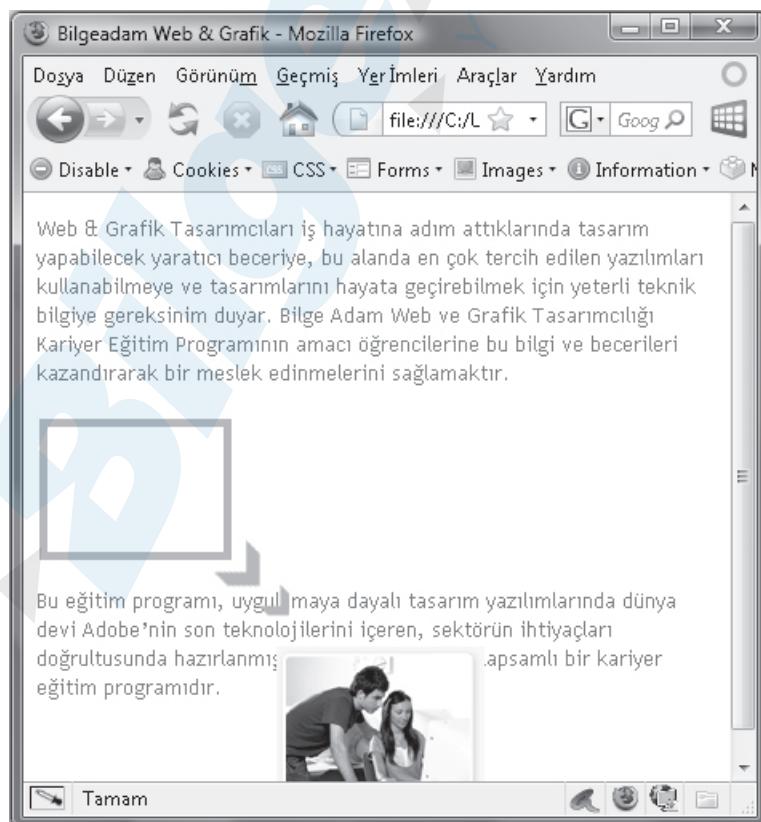
**Figür 8-b-3**

Görselimiz önceden bulunduğu pozisyon 0 noktası baz alınarak, soldan 150, yukarıdan da 150 piksel mesafeeye yerleşerek konumlandı.

Burada pozisyonunu ilk paragraftan sonra geldiği için, o paragraftan sonraki eski konumundan (tarayıcı akış konumu) başlayarak verdigimiz offset noktasına yerleşmesinden bahsediyoruz.

Bunun haricinde dikkat ettiyseniz, eski pozisyonundayken kapladığı alan, halen boş da olsa yer kaplamaya devam ediyor, yani ilk paragraftan sonra görselimizin bulunduğu hacim alanı halen mevcut.

Tarayıcımızı tekrar boyutlandıracak olursak da, görselimizin position özelliği uygulanmamış olan diğer elementlerin üstüne binen bir katman gibi yerleşmiş olduğunu da görürüz:

**Görüntü 8-b-1**

Bağıl konumlandırma görüldüğü gibi beraberinde düşünülmesi gereken birçok sıkıntıyı getirebilir. Bu yüzden bağıl konumlandırmayı genelde position özelliği tanımlanmış başka elementlerle birlikte kullanıyoruz.

Bağıl konumlandırmayla ilgili bir başka örneğe daha sonra değineceğiz.

8.c Mutlak (Absolute) Konumlandırma

Bir önceki konuda, bağıl konumlandırmmanın, uygulanan elementin tarayıcı akış konumu baz alınarak ofset noktalarındaki pozisyonlara verilen değerlerle tekrar konumlandırma yapıldığından bahsetmiştim.

Bu kez elementimizi mutlak (absolute) bir pozisyon'a nasıl konumlandırabileceğimizi göreceğiz.

Örnek uygulaması için, bir önceki dokümanımızı img elementimizde pozisyon düzenlemesi olmadan tekrar açıyoruz:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title>Bilgeadam Web & Grafik</title>
<style type="text/css">
body {
font:9.5pt "Trebuchet MS"; /*doküman geneli yazıtipi özellikleri*/
color:#666666; /*gri metin rengi*/
}
</style>
</head>

<body>
<p>Web & Grafik Tasarımcıları iş hayatına adım attıklarında tasarım yapabilecek yaratıcı beceriye, bu alanda en çok tercih edilen yazılımları kullanabilmeye ve tasarımlarını hayatı geçirebilmek için yeterli teknik bilgiye gereksinim duyar. Bilge Adam Web ve Grafik Tasarımcılığı Kariyer Eğitim Programının amacı öğrencilerine bu bilgi ve becerileri kazandırarak bir meslek edinmelerini sağlamaktır.</p>



<p> Bu eğitim programı, uygulamaya dayalı tasarım yazılımlarında dünya devi Adobe'nin son teknolojilerini içeren, sektörün ihtiyaçları doğrultusunda hazırlanmış, toplam 500 saatlik kapsamlı bir kariyer eğitim programıdır. </p>

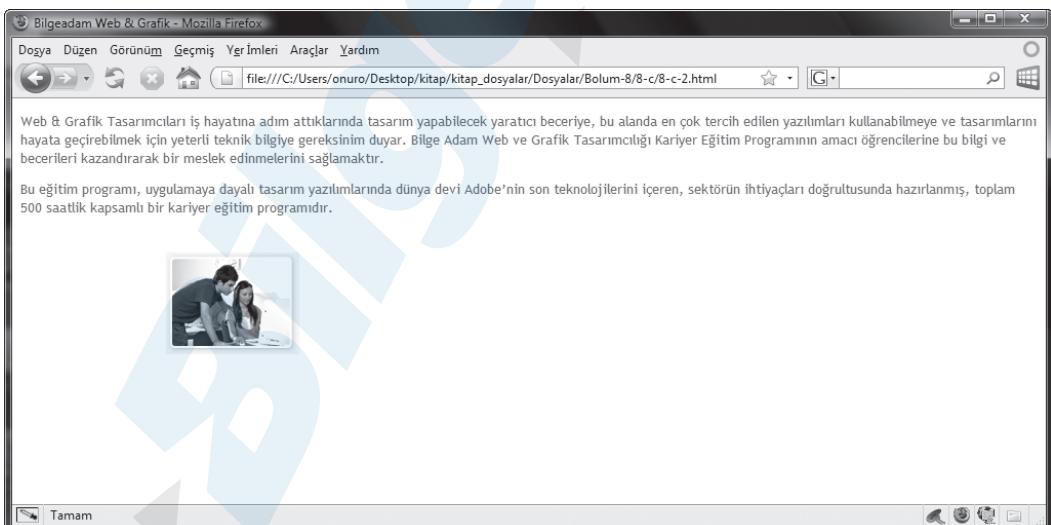
</body>
</html>
```

**Figür 8-c-1**

Şimdi yapacağımız işlem, görselimizi alıp mutlak, yani konumu hiçbir şekilde değiştirmeyen, sabit bir noktaya taşımak.

`position` özelliğine atayacağımız `absolute` değeri ile mutlak konumlandırmamız gerçekleşecek ancak, `absolute` konum değeri de `relative` gibi ofset noktalarına ihtiyaç duyar. Dolayısıyla sol ve üstten bir önceki değerlerle aynılarını, 150 piksel ofset mesafesini uygulayalım:

```
img {
    position: absolute;
    top: 150px;
    left: 150px;
}
```

**Figür 8-c-2**

Relative konumlandırma örneğimizle karşılaşırıracak olursak, absolute konumlandırma uyguladığımızdaki konum durumu relative ile oldukça benzeşiyor. Ancak burada bazı farklar var.

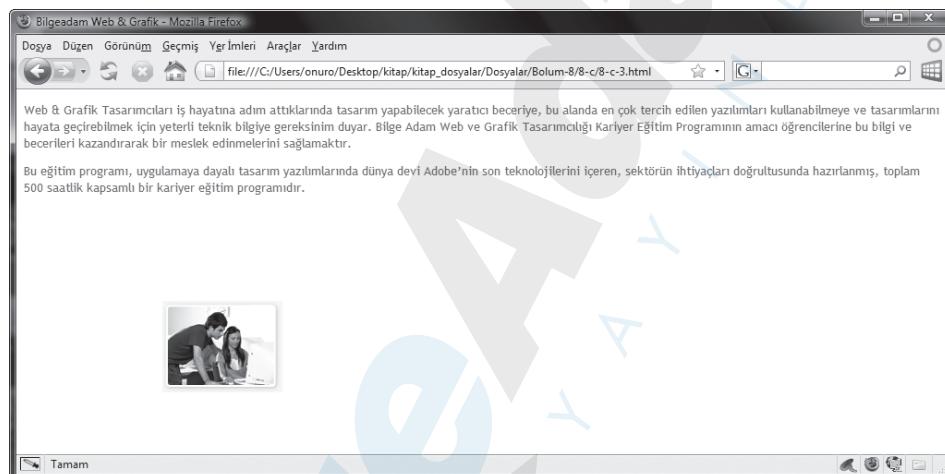
Absolute konumlandırmada, ofset noktası verilen element tarayıcı akış konumunda baz olarak eski konumunu değil, bir üst elementinin 0 noktasını baz alır. Bu örneğimizde bir üst elementimiz `body` elementidir (bu örnekte `img` öğesini kapsayan daha büyük başka bir element yoktur).

Bir başka dikkat çeken nokta da, relative konumlandırmada varlığını koruyan eski hacim boşluğu, absolute konumlandırmada ortadan kalkıyor.

Bu şu anlama gelir: **Absolute konumlandırma uygulanan element eski bulunduğu konumdan kopar ve diğer öğelerden bağımsız bir yerleşime sahip olur.** Bu vesileyle, absolute konumlandırma uyguladığımız bir öğeyi diğer elementlerden bağımsız olarak doküman içerisinde istediğimiz noktaya yerleştirip orada sabitleyebiliriz.

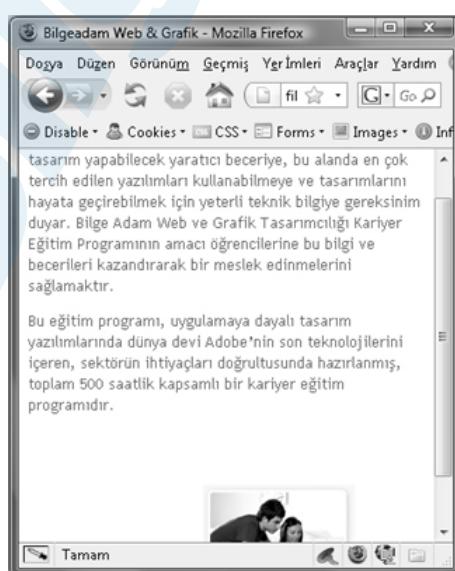
Bazen absolute konumlandırma uyguladığımız öğeyi son konumundan tamamen koparmadan yarı-bağımsız yerleşime sokmak isteyebiliriz. Böyle bir durumda, öğeyi bir üst elementine veya tarayıcı akış konumuna oranla belirli bir mutlak konuma ofset noktaları yerine margin boşluklarıyla da taşıyabiliriz:

```
img {
    position: absolute;
    margin-top: 150px;
    margin-left: 150px;
}
```



Figür 8-c-3

Fark edeceğiniz üzere, tarayıcılarda ofset noktalarının piksel değerleriyle üst element bağlantı pozisyonları ile margin boşluklarının mesafe oranları farklı yorumlanır.



Görüntü 8-c-1

Margin boşlukları ile absolute konumlandırmayı, öğelerimizi position özelliği atanmamış başka elementlere uyumlu ve hizalı bir şekilde yerleştirmek için kullanabiliriz. Böyle bir durumda konumlanmanın, ögenin bir üst elementinin başlangıç noktasına olan mesafesi baz alınarak gerçekleşeceğini hatırlamamız yeterlidir.

Şu ana kadar bahsettiğlerim size biraz karmaşık gelebilir (evvelce öğrenirken bana da öyle gelmişti). Bol deneme ve uygulama yaparak, iki pozisyon değerine hakimiyet kapasitenizi artırarak birçok yaratıcı konumlandırma uygulaması yapabilirsiniz.

Nitekim bir sonraki bölümde, iki konumlandırma değerinin birlikte kullanımına yönelik bir uygulama örneği yapacağız.

8.d Hizalanmış İçerik Düzeni İçin Relative ve Absolute Konumlandırmayı Birlikte Kullanmak

position özelliğinin doğru kullanımının biraz teknik pratiği gerektirdiğinden bahsetmiştim.

Bu bölümümüzde, position özelliğini bazı elementlere farklı değerlerle atayarak konumlandırma konusunu bizim için nasıl biraz daha kolay hale getirebileceğimizi göreceğiz.

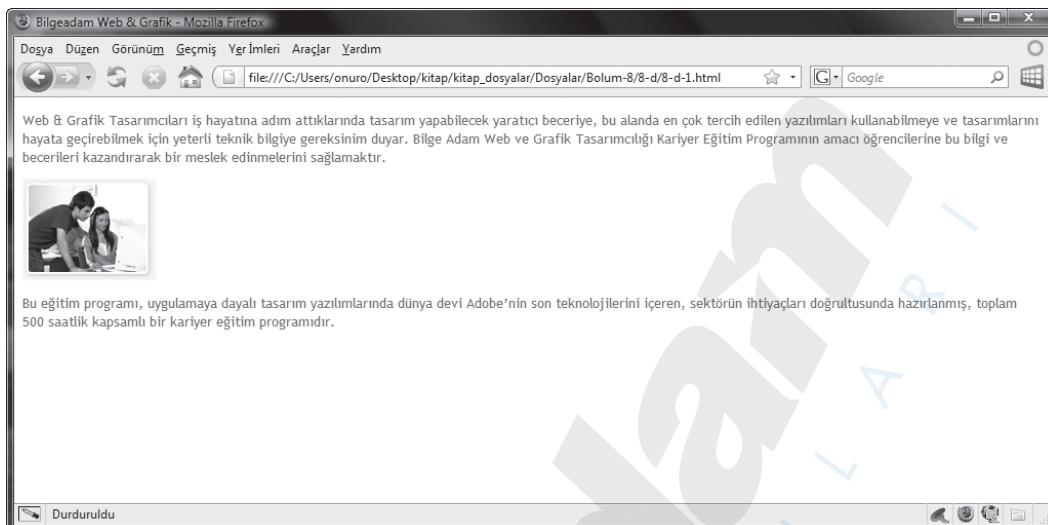
Öncelikle, diğer örneklerimizde kullandığımız dosyayı, paragrafları id selektörüne sahip bir div kutusu ile kapsayarak, standart stilleriyle tekrar açalım:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title>Bilgeadam Web & Grafik</title>
<style type="text/css">
body {
font:9.5pt "Trebuchet MS"; /*doküman geneli yazıt tipi özellikleri*/
color:#666666; /*gri metin rengi*/
}
</style>
</head>

<body>
<div id="icerik">
<p>Web & Grafik Tasarımcıları iş hayatına adım attıklarında tasarım yapabilecek yaratıcı beceriye, bu alanda en çok tercih edilen yazılımları kullanılmaya ve tasarımlarını hayatı geçirebilmek için yeterli teknik bilgiye gereksinim duyar. Bilge Adam Web ve Grafik Tasarımcılığı Kariyer Eğitim Programının amacı öğrencilerine bu bilgi ve becerileri kazandırarak bir meslek edinmelerini sağlamaktır.</p>

<p> Bu eğitim programı, uygulamaya dayalı tasarım yazılımlarında dünya devi Adobe'nin son teknolojilerini içeren, sektörün ihtiyaçları doğrultusunda hazırlanmış, toplam 500 saatlik kapsamlı bir kariyer eğitim programıdır. </p>
```

```
</div>
</body>
</html>
```



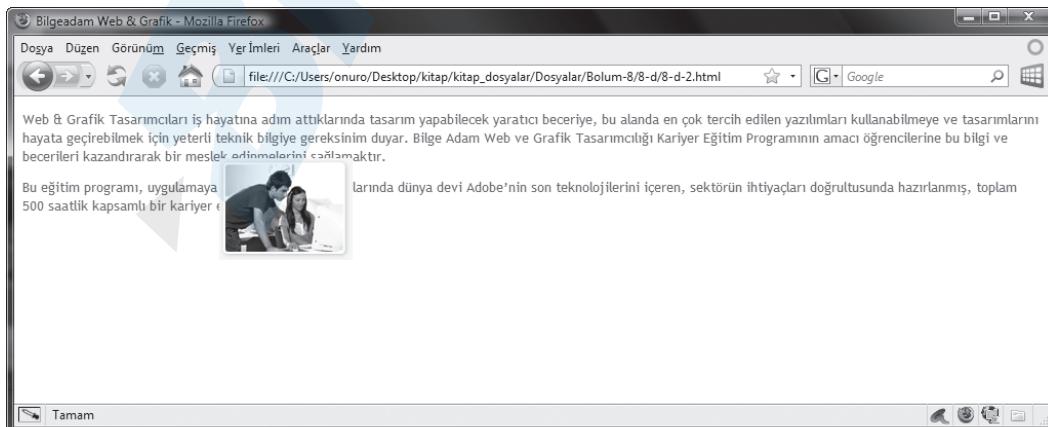
Figür 8-d-1

Önceki örneklerimizde görsel elementimize uyguladığımız pozisyon özelliklerini, görselimizin ya diğer öğelerden koparak bağımsız hareket etmesine, ya da dokümanın genel akış konumlandırmasına göre hizalanıp muhtemel görünüm sıkıntılıları yaratabilecek durumlara neden olmuştu.

Bu kez hesaplarımıza doğru yapıp, görsel elementimizi div kutusunun oranlarına göre konumlandıracıcağız, ve tüm tarayıcılarda ortak görünüm sahip olacak bir sabit konumlandırma uygulayacağız.

Öncelikle, görselimizin konumunu mutlak değere sahip bir noktaya taşıyacağımız için ofset noktalarını belirterek absolute konumlandırmasını gerçekleştiriyoruz:

```
img {
    position: absolute;
    top: 60px;
    left: 200px;
}
```



Figür 8-d-2

Görselimiz bulunduğu eski konumdan koparak 60 piksel aşağı, 200 piksel de sola kaymış oldu. Burada görselimizin eski konumdan kopmasından bahsederken, **icerik** div kutusu içindeki haliyle de bir bağlantısının kalmadığından, tamamen bağımsız bir konuma yerleştiğinden bahsediyoruz.

Şimdi bu bahsettiklerimden emin olmak ve **icerik** div kutumuzu biraz daha belirginleştirmek adına **icerik** id selektörümüze biraz iç boşluk, yukarıdan biraz dış boşluk ve çerçeve eklemek amaçlı bazı tanımlamalar yapalım:

```
#icerik {
    margin-top:150px;
    padding:10px;
    border:1px dotted #ccc;
}
```



Figür 8-d-3

Göründüğü gibi, görselimiz **icerik** div kutusundan tamamen bağımsız bir şekilde mutlak bir pozisyon'a konumlanmış durumda.

Bu durumu değiştirmek, görselimizin mutlak konumunu **icerik** kutumuzun oranlarına göre yerleşmesini sağlamak, dolayısıyla bu kutuya bağlantı kurulabilmesini sağlamak için **icerik** kutumuzu bağlı bir konum değeri vermemiz gerekiyor.

Bu nedenle **icerik** kutumuzun pozisyon değerine relative uygulaması yaparsak, görsel ve kutu ilişkisi gerçekleşecektir:

```
#icerik {
    position:relative;
    margin-top:150px;
    padding:10px;
    border:1px dotted #ccc;
}
```



Figür 8-d-4

Kutumuz ile görselimizin bağlantısını sağlayarak, görselimizin mutlak bir konuma yerleşmesini sağlamış olduk. Bu konumlandırma teknikleriyle yapabileceğimiz görsel uygulamaların sınırı olmayacağı olacaktır.

position özelliğini kullanarak konumlandırma konumuzun başında da söylediğim gibi, normal şartlarda position özelliğine zaruri bir ihtiyacımız olmayacak. Ancak ekseri durumlarda, uygunladığımız örneklerdeki gibi belirli noktalara belirli elementleri sabit konumlarla yerleştirmemiz gereken durumlarda position özelliği bize oldukça yardımcı olacak.

8.e Sabit (Fixed) Konumlandırma

Bağıl ve mutlak konumlandırmada, seçtiğimiz bir elementi dilediğimiz bir ofset noktasına serbest bir şekilde, diğer doküman elementlerin konumlarıyla bağlantılı bir şekilde yerleştirilmesini görmüşük.

Bazen, konumlandırılacağımız öğeyi dokümanda belli bir noktaya, sayfa scroll etse dahi konumunu koruyacak şekilde yerleştirmek isteyebiliriz.

Ne bağıl konumlandırma, ne de mutlak pozisyonaya yerleştirme ne de başka bir CSS yöntemi bizim bu dileğimizi gerçekleştirecektir.

Öğelerimizi dokümanımızda belli bir konuma, durumları sabit kalacak şekilde yerleştirebilmek için position özelliği için fixed değerini kullanacağız. Bu değer uygulanmış öğeler, tüm öğelerden bağımsız sabit bir noktaya yerleştirilebilir ve doküman scroll etse dahi o noktadaki pozisyonlarını korumaya devam edebilirler.

Şimdi uygulayacağımız örnekte, bir önceki dokümanımızı tekrar açalım ve görselimi belli bir noktaya sabitlemeye çalışalım.

Ben dokümanda biraz scroll oluşması için varolan paragrafları birkaç kez tekrar ettiriyorum:

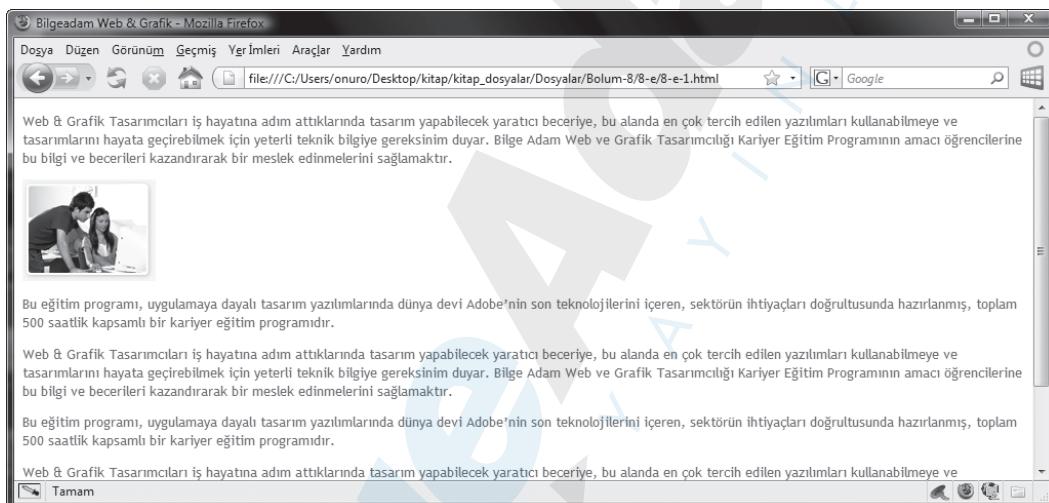
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title>Bilgeadam Web & Grafik</title>
<style type="text/css">
```

```
body {  
font:9.5pt "Trebuchet MS"; /*doküman geneli yazıt tipi özelliklerini*/  
color:#666666; /*gri metin rengi*/  
}  
</style>  
</head>  
<body>  
<div id="icerik">  
  
<p>Web & Grafik Tasarımcıları iş hayatına adım attıklarında tasarım yapabilecek yaratıcı beceriye, bu alanda en çok tercih edilen yazılımları kullanabilmeye ve tasarımlarını hayatı geçirebilmek için yeterli teknik bilgiye gereksinim duyar. Bilge Adam Web ve Grafik Tasarımcılığı Kariyer Eğitim Programının amacı öğrencilerine bu bilgi ve becerileri kazandırarak bir meslek edinmelerini sağlamaktır.</p>  
  
  
<p> Bu eğitim programı, uygulamaya dayalı tasarım yazılımlarında dünya devi Adobe'nin son teknolojilerini içeren, sektörün ihtiyaçları doğrultusunda hazırlanmış, toplam 500 saatlik kapsamlı bir kariyer eğitim programıdır. </p>  
  
<p>Web & Grafik Tasarımcıları iş hayatına adım attıklarında tasarım yapabilecek yaratıcı beceriye, bu alanda en çok tercih edilen yazılımları kullanabilmeye ve tasarımlarını hayatı geçirebilmek için yeterli teknik bilgiye gereksinim duyar. Bilge Adam Web ve Grafik Tasarımcılığı Kariyer Eğitim Programının amacı öğrencilerine bu bilgi ve becerileri kazandırarak bir meslek edinmelerini sağlamaktır.</p>  
  
<p> Bu eğitim programı, uygulamaya dayalı tasarım yazılımlarında dünya devi Adobe'nin son teknolojilerini içeren, sektörün ihtiyaçları doğrultusunda hazırlanmış, toplam 500 saatlik kapsamlı bir kariyer eğitim programıdır. </p>  
  
<p>Web & Grafik Tasarımcıları iş hayatına adım attıklarında tasarım yapabilecek yaratıcı beceriye, bu alanda en çok tercih edilen yazılımları kullanabilmeye ve tasarımlarını hayatı geçirebilmek için yeterli teknik bilgiye gereksinim duyar. Bilge Adam Web ve Grafik Tasarımcılığı Kariyer Eğitim Programının amacı öğrencilerine bu bilgi ve becerileri kazandırarak bir meslek edinmelerini sağlamaktır.</p>  
  
<p> Bu eğitim programı, uygulamaya dayalı tasarım yazılımlarında dünya devi Adobe'nin son teknolojilerini içeren, sektörün ihtiyaçları doğrultusunda hazırlanmış, toplam 500 saatlik kapsamlı bir kariyer eğitim programıdır. </p>
```

```
<p>Web & Grafik Tasarımcıları iş hayatına adım attıklarında tasarım yapabilecek yaratıcı beceriye, bu alanda en çok tercih edilen yazılımları kullanabilmeye ve tasarımlarını hayatıya geçirilemeye için yeterli teknik bilgi gereksinim duyar. Bilge Adam Web ve Grafik Tasarımcılığı Kariyer Eğitim Programının amacı öğrencilere bu bilgi ve becerileri kazandırarak bir meslek edinmelerini sağlamaktır.</p>
```

```
<p> Bu eğitim programı, uygulamaya dayalı tasarım yazılımlarında dünya devi Adobe'nın son teknolojilerini içeren, sektörün ihtiyaçları doğrultusunda hazırlanmış, toplam 500 saatlik kapsamlı bir kariyer eğitim programıdır. </p>
```

```
</div>
</body>
</html>
```

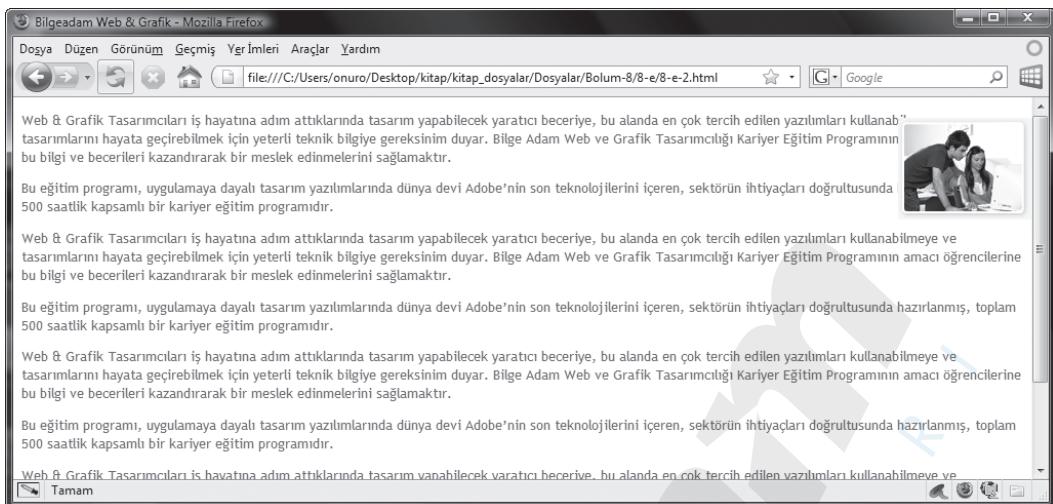


Figür 8-e-1

Şimdi yapacağımız işlem, görselimizi dokümanın en sağına yerleştirip tepeden 20 piksellik boşluk bırakarak o pozisyonu sabitlemek.

Sabitleme için position özelliğine fixed değeri verirken, konum belirtmek için tekrar ofset noktalarına ihtiyaç duyacağız:

```
img {
  position:fixed;
  right:0;
  top:20px;
}
```

**Figür 8-e-2**

Gördüğü gibi, görselimiz eski konumundan koparak belirttiğimiz pozisyon'a sabitlenecek şekilde yerleştı.

İlk etapta float uygulanmış gibi görünse de, diğer elementlerin üstünde olduğunu ve sabit kaldığı-nı dokümanımızı scroll ettirdiğimizde daha iyi görürüz:

**Görüntü 8-e-1**

Oldukça pratik ve etkileyici, değil mi?

CSS ile position özelliği üzerinde denemeler ve uygulamalar yaparak kendinizi konumlandırma hususunda daha da geliştirebilir, yaratıcı tasarım fikirlerinizi Web dokümanlarınızda doğrudan hayata geçirilebilirsiniz.



9

CSS ile Form Öğelerini Stillendirmek

9 CSS ile Form Öğelerini Stillendirmek

- (x)HTML ile Erişilebilir Formlar Hazırlamak
- CSS ile Fieldset İçindeki Öğeleri
Stillendirmek
- CSS ile Görsel Fieldset Izgarası (Grid)
Oluşturmak-1
- CSS ile Görsel Fieldset Izgarası (Grid)
Oluşturmak-2
- Form Butonlarını Stillendirmek

CSS ile Form Öğelerini Stillendirmek

Eskiden bu yana, tasarladığımız Web sitelerinde teknik yetersizlik nedeniyle sınırlı seviyede görseleştirmeler yaparak hayatı geçirdiğimiz öğeler arasında kuşkusuz ki interaktif formlar devamlı başı çekmiştir.

Bunun sebebi, tarayıcıların form etiketinin altında bulunan form öğelerinin (text kutusu, radyo butonları, liste, menüler vb.) işletim sistemi standardında görüntülenmeleri ve o vakitler CSS ile tasarım geliştirmek sınırlar dahilinde olduğu için birçok kısıtlamaya sahip olmamızdı.

CSS ile bu sıkıntıların bir çoğunu aşarak stilsiz halleriyle (üzgünüm ama doğrudan söylemek zorundayım) son derece çirkin ve kullanışsız görünen formlarımızı daha şık ve erişilebilir hale getireceğiz.

Bu bölümde uygulayacağımız örneklerde bir interaktif iletişim formunu baz alacağız ve formun öğelerini sırayla biçimleyerek daha göze hitap eden bir yapıya kavuşturacağız.

Bu işlemlere geçmeden önce, hatırlamamız gereken bir konu var: Dokümanlarımız, dolayısıyla içindeki form öğeleri de dahil olmak üzere tüm içeriği farklı platformlardan da erişilebilir olmalıdır.

Dolayısıyla ilk aşamada (x)HTML ile erişilebilir form sayfaları geliştirmeye yönelik bir örnek yapacağız.

9.a (x)HTML ile Erişilebilir Formlar Hazırlamak

Her ne kadar aranızda (x)HTML diline hakim olanlarınızın formlar oluşturma hususunda bilgi sahibi olduğunu tahmin etsem de, konunun detayına inip stillendirme işlemlerimizi daha doğru bir şekilde yapmak adına erişilebilir form oluşturmaya değinmek istiyorum.

Bilindiği gibi, formlar kendilerini kapsayan ve interaktif veri transferi işleminde köprü görevi üstlenen kutu seviyesi `<form>` etiketiyle kapsanırlar.

Bu etiketlerin içine giren öğelerin doğrudan stillendirilmesi ve kutu yapısında biçimlenmeleri birçok durumda tasarımlarımızı hayatı geçirirken yeterli gelmeyeceği için, formlarındaki öğeleri gruplamamızı ve adımlara bölmemizi sağlayan bir başka kutu seviyesi element olan `fieldset` elementlerini kullanıyor olacağız.

Hazırlayacağımız iletişim Formumuzda veri giriş kriterleri şu şekilde olsun:

- Ad
- Soyadı
- E-mail
- Mesaj

Oluşturacağımız xHTML dokümanının yapısal düzenini aşağıdaki gibi hazırlayacağız:

1. Kriterlerimizden ilk üçünü bir `fieldset` alanı içine, mesaj bölümünü de bir başka `fieldset` içine alarak mantıksal bir gruplama yapacağız.

Bu yöntem hem stillendirme yaparken daha yaratıcı fikirleri hayatı geçirmemize yardımcı olacak, hem de ziyaretçinin bir sürü metin kutusu arasında kaybolmaması adına bir iyileştirme sağlayacaktır.

2. Veri kutularımızı her zamanki gibi `input` ile oluştururken, birden fazla satır sahip olabilecek mesaj metni gibi öğelerimiz için `textarea` form öğesini kullanacağız.

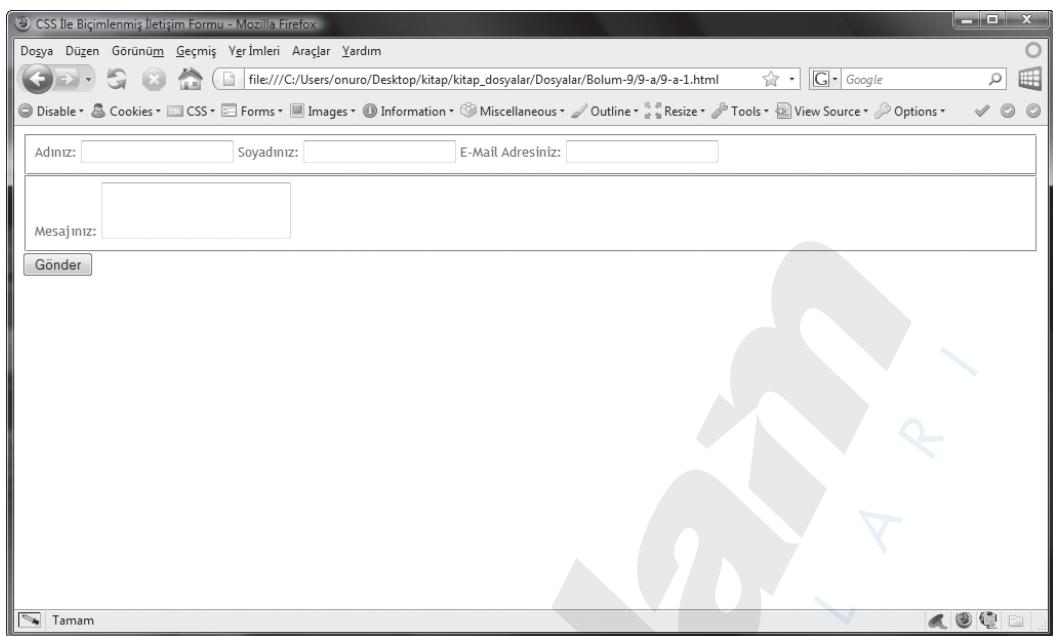
3. Kriter başlıklarımızın ve onların veri giriş içeriklerinin erişilebilir olması için de onları label elementiyle kapsayacağız.

Şimdi dilerseniz bu bahsettiklerimizi uygulamaya koyalım ve bu işlemi yaparken dokümanımızda genel bazı görünüm düzenlemeleri de ekleyelim:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title>CSS İle Biçimlenmiş İletişim Formu</title>
<style type="text/css">
    body {
        font:9.5pt "Trebuchet MS";
        color:#666666;
    }
</style>
</head>
<body>
<form id="iletisim" method="post" action="">
<fieldset>
    <label for="ad"><span>Adınız:</span>
    <input type="text" name="ad" id="ad" /></label>
    <label for="soyad"><span>Soyadınız:</span>
    <input type="text" name="soyad" id="soyad" /></label>
    <label for="e-mail"><span>E-Mail Adresiniz</span>
    <input type="text" name="email" id="email" /></label>
</fieldset>
<fieldset>
    <label for="mesaj"><span>Mesajınız:</span>
    <textarea name="mesaj" id="mesaj"></textarea></label>
</fieldset>
<div><input type="submit" value="Gönder" /></div>
</form>
</body>
</html>
```

label içine aldığımız kriter başlıklarını neden span ile kapsadığımızı ilerleyen aşamalarda göreceğiz.

Dokümanımızın görünümüne bir göz atalım:



Figür 9-a-1

Şu an itibarıyle genel bazı stillendirme işlemlerimize rağmen iletişim formumuz oldukça albeniden uzak ve itici görünüyor.

Önümüzdeki aşamada işe bu formdaki **fieldset** içinde bulunan öğelerin görünümünü şu andan kinden biraz daha sık hale getirmeye çalışacağız.

9.b CSS ile Fieldset İçindeki Öğeleri Stillendirmek

Bir önceki aşamada, çapraz tarayıcı uyumlu ve erişilebilir formları nasıl oluşturabileceğimizi görmüştük.

Şimdi bu formların içindeki öğelerin içerik kısmını stillendirerek yapısal biçimlemelere geçmeden önce onları görsel olarak bu biçimlemelere hazır hale getireceğiz.

Yapacağımız düzenlemeler, form içeriğimizi stilsiz standart halleri olan yan yana dizilim yapısından kurtarak başlık altına metin kutuları gelecek şekilde, düzgün bir görünümde sunmaktan ibaret olacak.

Yalnız burada dikkat etmemiz gereken konu, metin kutularımızın üstündeki başlık metinlerini span ile kapsamış olmamız.

Bu metinler ve yanlarındaki metin kutuları teoride kutu seviyesine sahip **label** etiketiyle kapsammış olsalar da, içeriğindeki span başlıklar kutu seviyesi değil satır içi (inline) yapıya sahip olduğundan, kutu yapısı elementler gibi kendileri float etmediği sürece yanlarına gelen elementi aşağıya atmıyorlar.

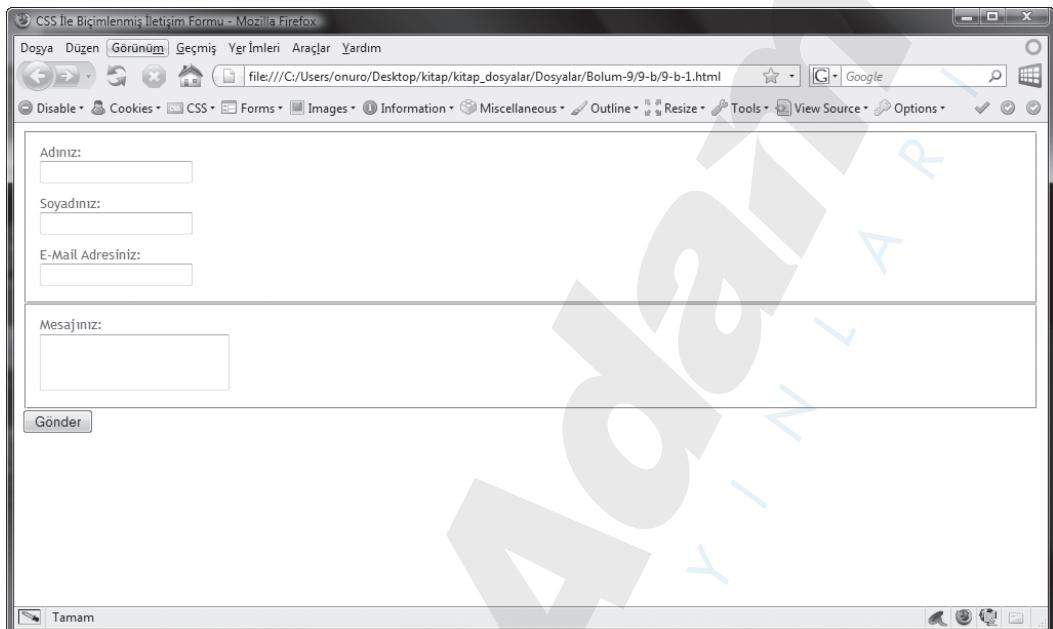
span öğelerimize kendilerine ait bir hacim kazandırmak için kutu seviyesi yapısına çevirirken (**display: block** işlemi), aynı işlemi teoride kutu seviyesi yapısına sahip **label** elementlerimizi de net olarak kutu seviyesi olduklarını tanımlamak için uygulayacağız.

Bu işlemleri yaparken sıkışıklığı ferahlatmak adına **label** elementlerimize biraz iç boşluk (**padding**) atayalım:

```

label, label span {
/* label ve label içindeki span öğeler için selektör graplama */
display:block;}
label {
padding:5px;
}

```

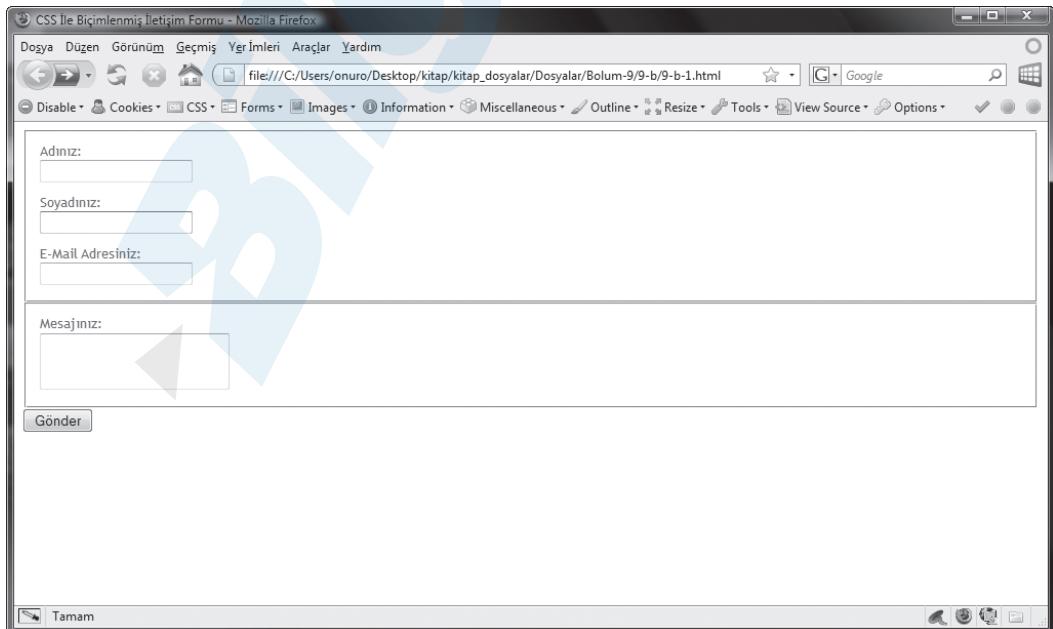


Figür 9-b-1

Az bir kod ve stillendirmeyle bile formumuzun görünümü stilsiz haline oranla ne kadar farklı oldu değil mi?

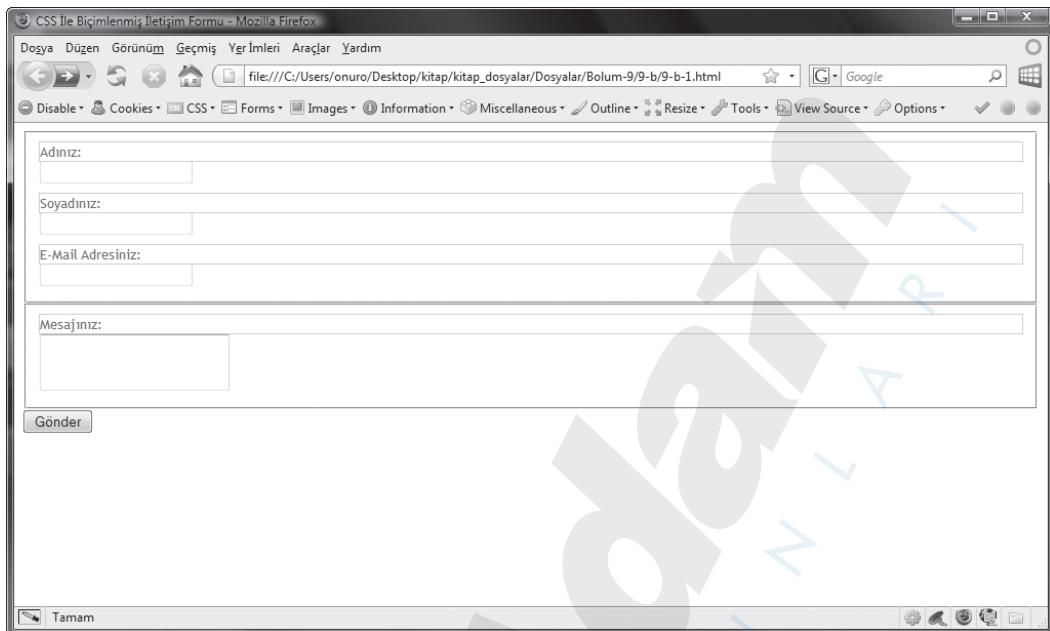
Burada bu değişikliğin en önemli aşamasını, `label` içindeki span öğelere, yani metin kutusu başlıklarımızı kapsayan satır içi span öğelere kutu seviyesi dönüştürmesi yaparak gerçekleştirdik.

Eğer span öğeleri kutu seviyesine dönüştürmeseydik, çalışmamız aşağıdaki gibi görünecekti:



Görüntü 9-b-1

Şunu hatırlamış olduk. Inline elementler, hacim sahibi nitelik taşımazlar ve o elementlerden sonra gelen öğeler yeni bir boşluk ya da satırda başlamazlar. Ancak biz CSS yardımıyla gereken her durumda kendilerini kutu yapısına çevirebiliyoruz (tersini de yapabiliyoruz):



Görüntü 9-b-2

Şimdi dilerseniz `fieldset` elementimiz içindeki `input` öğelerini biçimleyerek, metin girdisi yapılan kutularımızın görünümünde bazı değişikler yapalım.

Bu işlemi yaparken bir yandan başlıklarımızın görünümünü biraz daha belirginleştirmek için yazıtipini kalınlaştırarak güçlendiriyoruz (`font-weight` özelliği):

```
span {
    font-weight:bold;
}

fieldset input {
    color:#666;
    padding:2px;
    border:1px solid #a1a1a1;
}
```

The screenshot shows a Mozilla Firefox window displaying a contact form titled "CSS ile Biçimlenmiş İletişim Formu". The form consists of several input fields within a single `fieldset`. The fields are labeled "Adınız:", "Soyadınız:", "E-Mail Adresiniz:", and "Mesajınız:". Each label is followed by its respective input field. Below these is a "Gönder" (Send) button. The "Mesajınız:" field is a larger `textarea` element. The entire form is styled with CSS, including colors and borders.

Figür 9-b-2

Yapmaya çalıştığımız değişiklikleri gerçekleştirdik. Ancak fark ettiyseniz yalnızca `fieldset` içindeki `input` selektörüne uyguladığımız `stillendirmeler`, `input` sınıfına girmeyen kutu seviyesi `textarea` elementini etkilemedi.

Dolayısıyla gruba `textarea` elementini de dahil ediyoruz:

```
fieldset input, fieldset textarea {
    color:#666;
    padding:2px;
    border:1px solid #a1a1a1;
}
```

This screenshot shows the same contact form as in Figure 9-b-2, but with a different CSS rule applied. The rule `fieldset input, fieldset textarea` has been added to the existing styles. This rule targets both `input` and `textarea` elements within a `fieldset`. As a result, all form fields now have a consistent look, including the message area, which previously did not inherit the styles from the `input` selector.

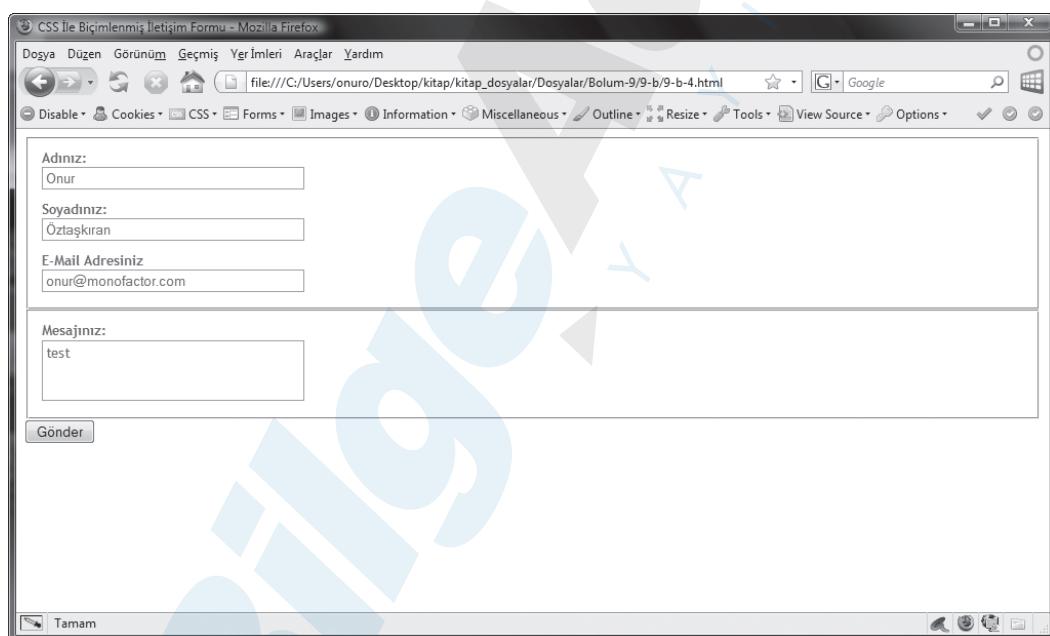
Figür 9-b-3

Bir sıkıntımız daha var: `textarea` elementini selektör grubumuza katarak ortak biçimlemeyi uygulamaya çalıştık ancak bunlara rağmen `textarea` yaztipinde bir farklılık var.

Bunun sebebi `textarea` elementinin metin özelliklerinin kendine has bir yapıya sahip olmasıdır. Dolayısıyla `input` elementleri ve diğer öğeler yaztipi özelliklerini üst seviye elementleri olan `body`'den kalıtsal olarak alırken, `textarea` bu kalıtsallığın dışında kalıyor.

Bu durumda `textarea` için yaztipi biçimlemelerini tekrar atamamız gerekiyor. Bu işlemi yaparken, ek olarak tüm metin kutularımıza **250** piksellik bir genişlik atayarak onlara sınır belirlemiş olalım:

```
fieldset input, fieldset textarea {
    color:#666;
    padding:2px;
    border:1px solid #a1a1a1;
    width:250px;
}
fieldset textarea {
    font:9.5pt "Trebuchet MS";
}
```



Figür 9-b-4

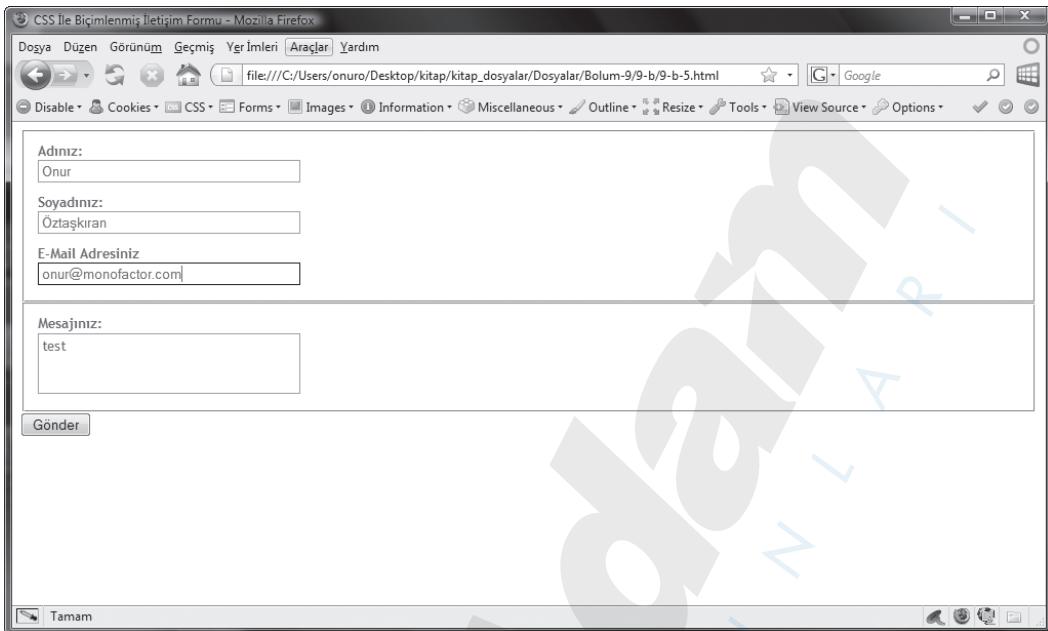
Artık tüm metin kutularımız ortak stil görünümüne sahip.

Son olarak, ziyaretçi fare işaretçisi ile metin kutusuna veri girişi yapmak için tıklayan ya da Tab tuşıyla metin kutularına erişerek yazmak istediginde kutularımızın görünümünde bazı değişimler uygulayarak formumuzun etkileyiciliğini artırmaya çalışacağız.

Bu bahsettiğim işlemler için daha evvelki uygulamalarımızda `a` elementi için de kullanmış olduğumuz **pseudo class'lardan** `:focus` özelliğini kullanacağız.

Ziyaretçi metin kutusuna veri girişi için tıkladığında kutularımızın çerçevesi rengini siyaha çevirelim:

```
fieldset input:focus, fieldset textarea:focus {  
border-color:#000;  
}
```



Figür 9-b-5

Böylece, `fieldset` elementlerimizin içeriğinin görünümünde son derece sık bir görünüm sağladık.

Bir sonraki aşamada, `fieldset` elementlerinin kendileri üzerinde değişiklik uygulayarak formumuzun görünüm iyileştirmesinde bir adım daha ileri gideceğiz.

9.c CSS ile Görsel Fieldset Izgarası (Grid) Oluşturmak-1

Son kaldığımız noktaya baktığımızda, dokümanımızdaki `fieldset` elementlerinin tarayıcı standartları görünümde 3 boyutlu etkisine çok yakın, aşağı yukarı 1-2 piksel genişliğinde gri çerçevelere sahip olduğunu görürüz.

Tarayıcılarda stilsiz haldeyken göreceğimiz tüm `fieldset` elementleri bu şekildedir. Biz bu görünümü ve daha fazlasını değiştirmek üzere `fieldset` elementlerimize stillendirmeler uygulayacağız.

Şimdi formumuzdaki `fieldset` elementlerinin çerçeve rengini ve aralarındaki boşluğu artırmaya çalışalım.

Ben bu örnekte `fieldset`'ler için çerçeve rengini 3 piksellik gri, boşluk oranını da her `fieldset` elementinden sonra dışa doğru alttan 15 piksel olacak şekilde ayarlıyorum:

```
fieldset {  
border:3px solid #999;  
margin-bottom:15px;  
}
```

The screenshot shows a Mozilla Firefox browser window displaying a contact form titled "CSS İle Biçimlenmiş İletişim Formu". The form consists of several input fields grouped by **fieldset** elements. Each **fieldset** has a thin black border. The fields include "Adınız:", "Soyadınız:", "E-Mail Adresiniz:", and "Mesajınız:". Below the message field is a "Gönder" button. The browser's toolbar and address bar are visible at the top.

Figür 9-c-1

fieldset alanlarımız belirginleşti.

Şimdi de, evvelce Photoshop ile hazırlanan bir arkaplan grafiğimizi (**fieldset.jpg**) yatay olarak tekrar edecek şekilde **fieldset** elementlerimizin arkaplanı olarak atayalım:

```
fieldset {
    border:3px solid #999;
    margin-bottom:15px;
    background:url(fieldset.jpg) repeat-x;
}
```

This screenshot shows the same contact form as Figür 9-c-1, but with a different visual style. The **fieldset** elements now have a light gray background with a horizontal blue and white striped pattern, which is the image from "fieldset.jpg" repeated across the width of the fieldset. The form fields and buttons are identical to the first version.

Figür 9-c-2

Başlangıçta pek kullanışlı olmayan formumuz uyguladığımız stillendirmeyle oldukça sık bir hale geldi.

Bir sonraki aşamada, **fieldset** kutularımızın yapıları üzerinde biraz daha düzenlemeye gideceğiz.

9.d CSS ile Görsel Fieldset Izgarası (Grid) Oluşturmak-2

CSS yardımıyla uyguladığımız stillendirmenin form öğelerimizin görünümünde ne kadar etkiliği olduğunu gördük.

Günümüzün popüler tasarım yöntemleri arasında en sık kullanılanlardan biri tasarımında düzgün hizalama için kullanıcı dostu hayatı izgaralar (grid) oluşturmaktır. Bu şekilde, aynı bir poster çalışması hazırlar gibi, öğelerin belli noktalara, öncelikleri sıralanmış halde ve odak noktaları belirlenecek şekilde yerleştirilmelerine dayanır.

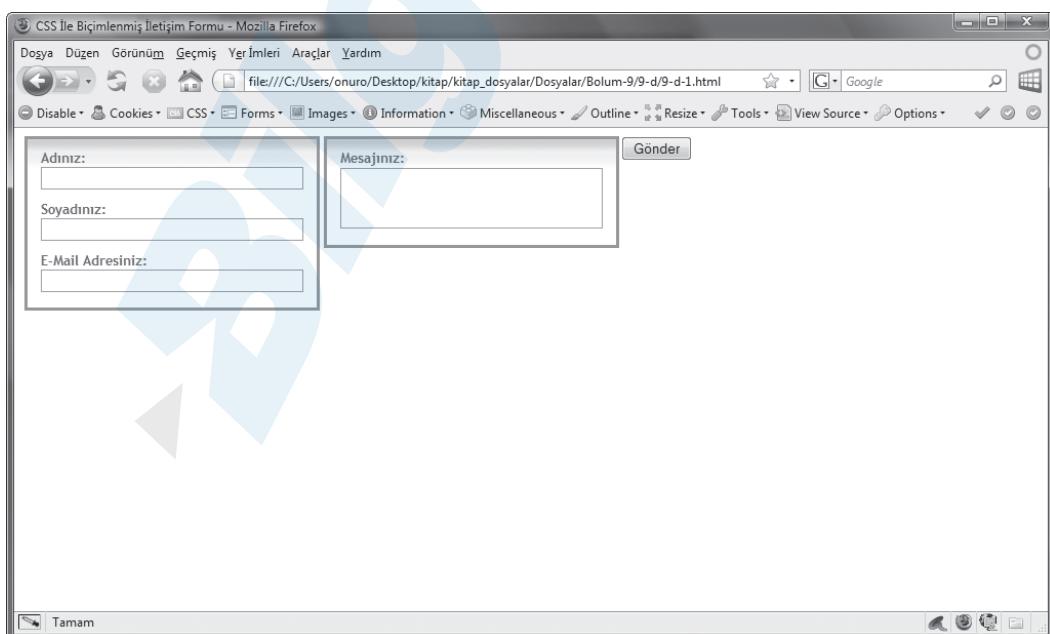
Biz bu mantığı, en temel seviyede de olsa iletişim formumuz için uygulamaya çalıştık.

Şimdiki aşamada, izgara görünümü oluşturmak için formumuz içindeki **fieldset** öğelerin hizalarını ve görünüm yapılarını biraz daha biçimlendirerek form görünümümüzü biraz daha zenginleştireceğiz.

Öncelikle **fieldset** öğelerini sola yaslayarak yan yana gelmelerini sağlayalım ve onlara 266 piksel değerinde birer genişlik atayalım:

```
fieldset {
    border:3px solid #999;
    margin-bottom:15px;
    background:url(fieldset.jpg) repeat-x;

    float:left;
    width:266px;
}
```



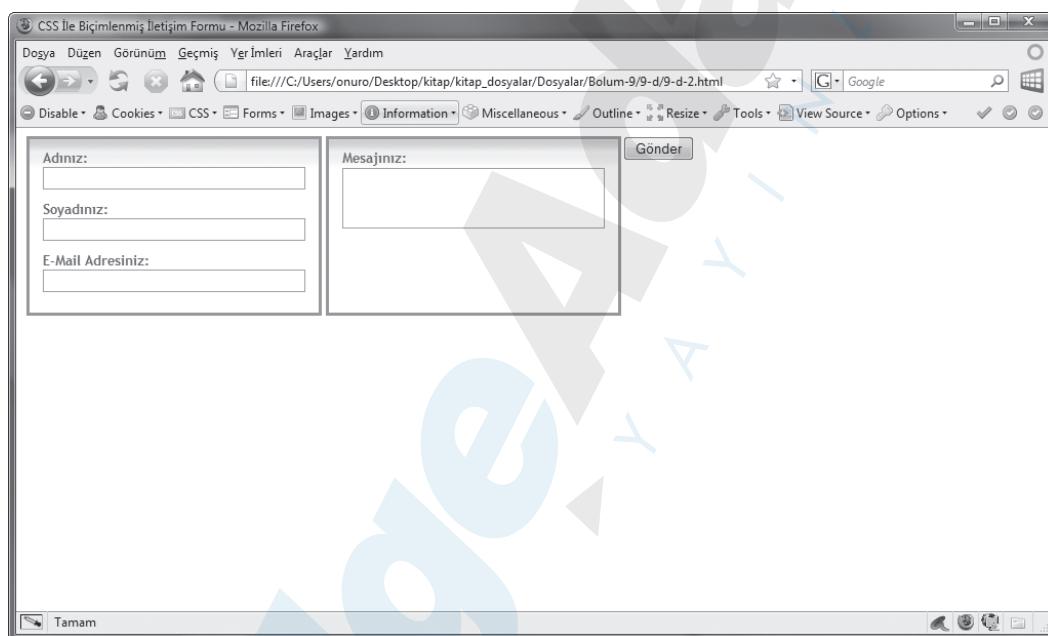
Figür 9-d-1

fieldset kutularımızın belirli bir genişlikle yan yana gelmelerini sağladık. Ancak henüz yükseklik uygulamadığımız için, her fieldset içeriği kadar yüksekliğe sahip. Bu da görünümdeki estetiğe etki ediyor.

fieldset öğelerimize sabit bir yükseklik uygulayalım:

```
fieldset {
border:3px solid #999;
margin-bottom:15px;
background:url(fieldset.jpg) repeat-x;

float:left;
width:266px;
height:155px;
}
```

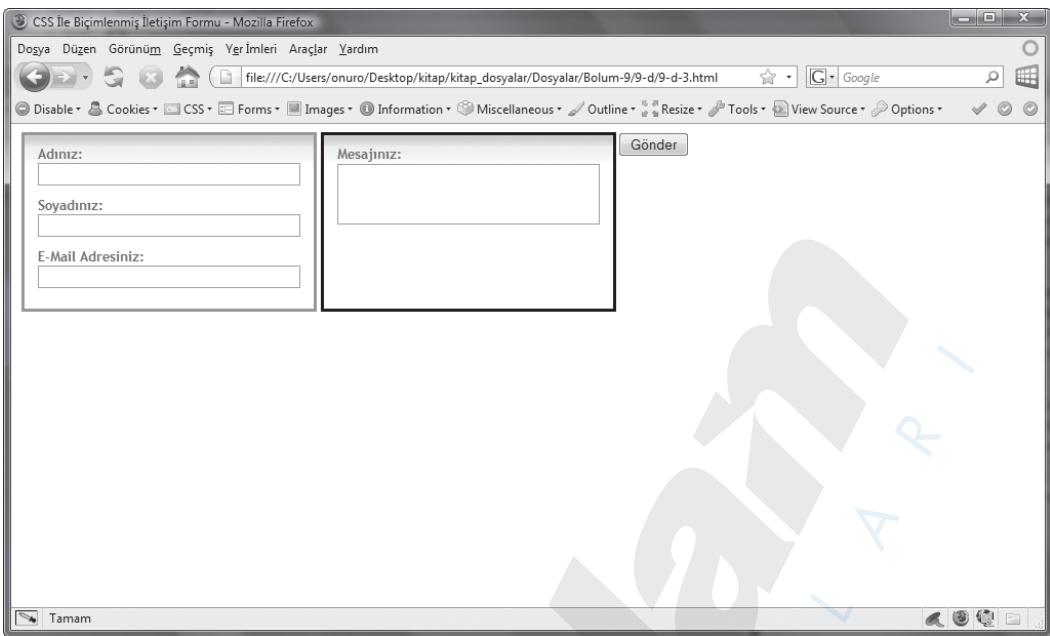


Figür 9-d-2

Böylece bir nebze hizalı ve uyumlu form kutularımızın bulunduğu bir dokümanımız oldu. Butonumuzun bulunduğu div şu an itibarıyle doküman genişliğimizden ve float eden fieldset öğelerimizden ötürü form öğelerinin hemen yanında durarak görünüm sıkıntısı yaratıyor.

Butonumuzun div'ini az sonra biçimlendireceğiz. Ondan evvel, fieldset öğelerimize hover durumunda çerçeveyi renginin değişimini uygulayarak biraz daha etkileşim katalım:

```
fieldset:hover {
border-color:black;
}
```



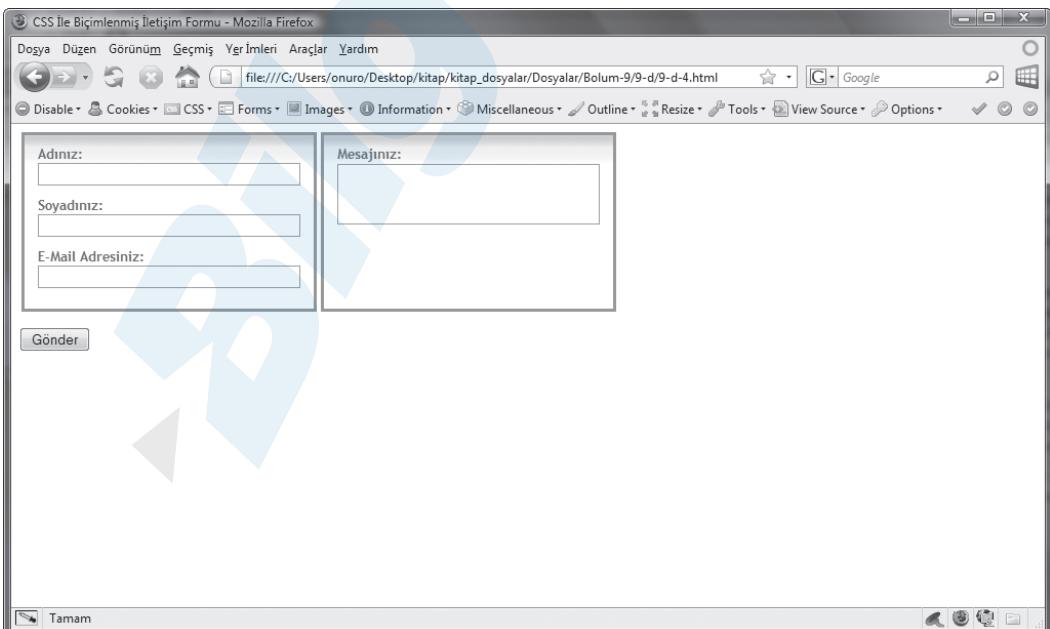
Figür 9-d-3

fieldset öğelerimizi ve içeriğini oldukça çekici bir hale getirmiş olduk.

Şimdi butonumuzun bulunduğu div elementini ait olduğu yere, fieldset öğelerimizin hemen altına getirelim. Bu işlem için, div elementimizin sol tarafını temizleyerek aşağı kaydılmamızı sağlayacak clear özelliğine ihtiyacımız var. clear özelliğini kurtarıcı elementler oluştururken görmüşük.

Formumuz içindeki div elementimizin sol tarafını temizliyoruz:

```
form div {
    clear:left;
}
```

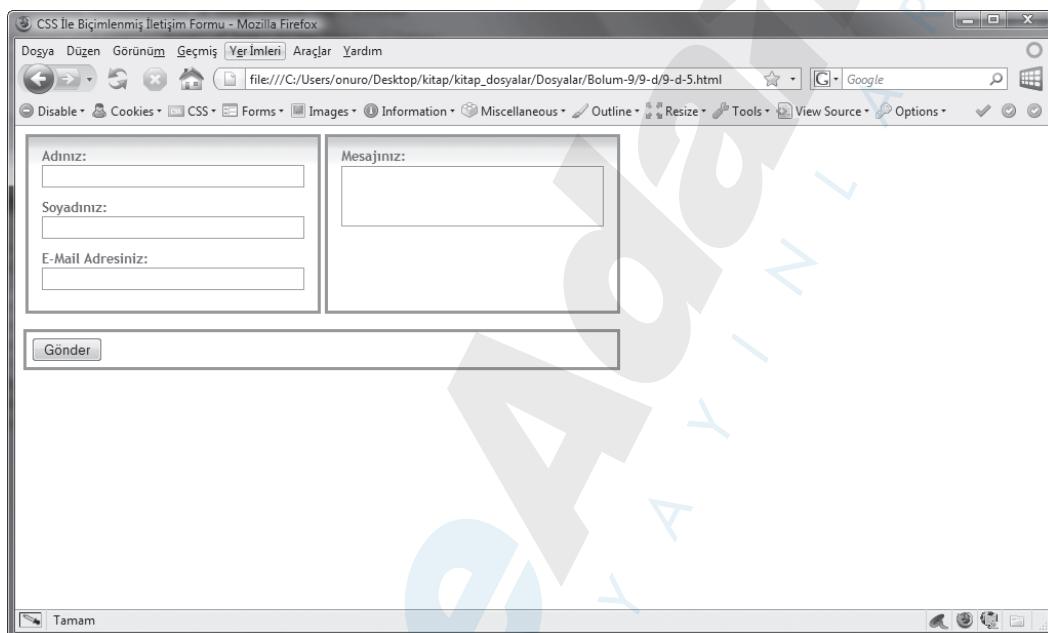


Figür 9-d-4

Oldukça düzgün bir yerleşime sahip olduk.

Şimdi dilerken son olarak butonumuzun bulunduğu div kutusuna fieldset için uyguladıkları-mızza benzer stillendirmeler uygulayalım:

```
form div {
    clear:left;
    width:566px;
    border:3px solid #999;
    padding:5px;
}
```



Figür 9-d-5

İşimiz neredeyse bitti gibi.

Ancak dikkat ederseniz div kutumuz 2 piksel içinde kalıyor gibi. Bu durumda kendisine soldan 2 piksellik boşluk verirken, genişliğini koruması için genişlik değerinden de 2 piksel kismamız gerekecektir:

```
form div {
    clear:left;
    width:564px;
    border:3px solid #999;
    padding:5px;
    margin-left:2px;
}
```

The screenshot shows a Mozilla Firefox window with the title "CSS İle Biçimlenmiş İletişim Formu - Mozilla Firefox". The address bar displays the local file path: "file:///C:/Users/onuro/Desktop/kitap/kitap_dosyalar/Dosyalar/Bolum-9/9-d/9-d-6.html". The main content area contains a form with three input fields labeled "Adınız:", "Soyadınız:", and "E-Mail Adresiniz:". Below these fields is a "Gönder" button. The entire form is contained within a single `fieldset` element.

Figür 9-d-6

Artık ilk görünümü kıyaslayacak olursak oldukça çekici bir iletişim formumuz var.

Bir sonraki bölümde, butonumuzun üzerinde biçimlemeler yapacağız.

9.e Form Butonlarını Stillendirmek

Hazırlamış olduğumuz iletişim formunun görünümü, standart buton görünümü haricinde neredeyse tamamlandı.

Form butonları, her tarayıcıda işletim sisteminin standardına göre görüntülenir. Buna rağmen elbette aynı `fieldset` öğelerinde olduğu gibi, buton elementlerimizi de CSS yardımıyla kolayca stillendirebiliyoruz.

Öncelikle form elementimizin kapsadığı `div` kutumuz içindeki buton `input` ögesi için bir selektör açalım ve ilk olarak yazıtipiyle arkaplan ve metin rengini değiştirelim:

```
form div input {
    color:#CCCCCC;
    background:#666666;
}
```

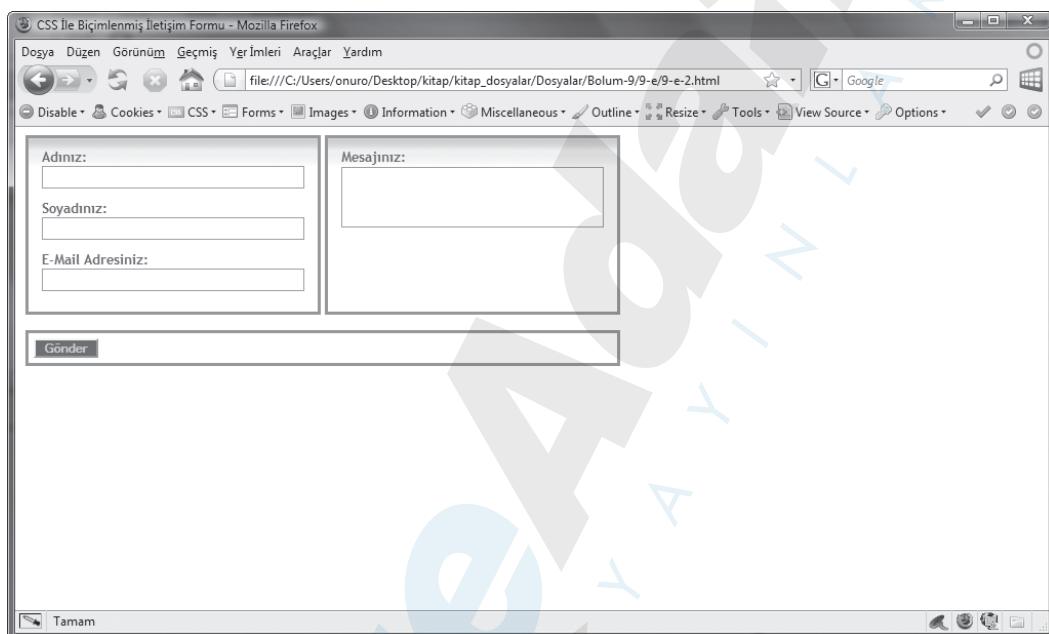
The screenshot shows the same Mozilla Firefox window as Figür 9-d-6, but with a styled "Gönder" button. The button now has a dark blue background color (#666666) and white text color (#CCCCCC). The rest of the form and its layout remain unchanged.

Figür 9-e-1

Nispeten daha belirgin bir buton haline geldi.

Şimdi dilerseniz butonumuz içindeki metnin boyutunu varolandan biraz daha ufak belirleyerek kalınlaştıralım:

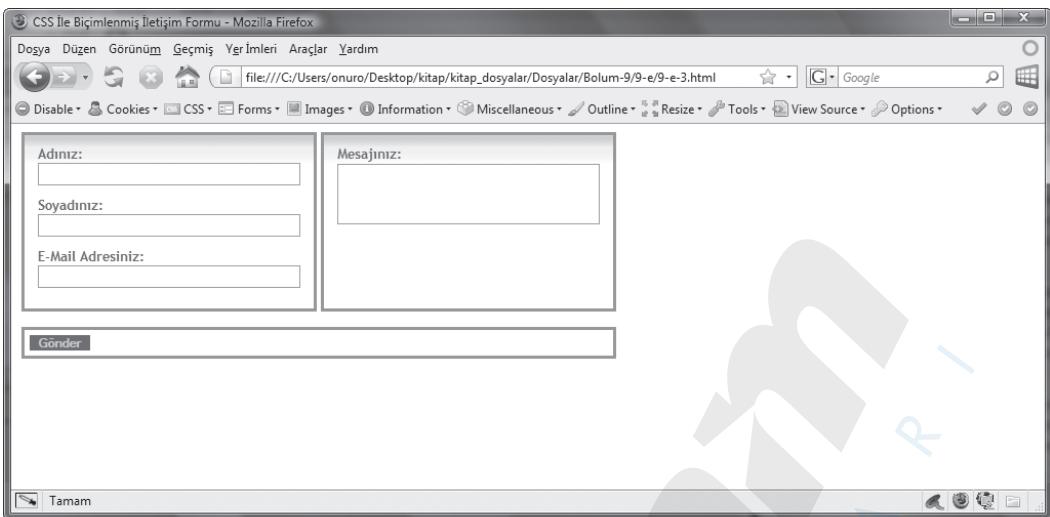
```
form div input {
    color:#CCCCCC;
    background:#666666;
    font-weight:bold;
    font-size:0.9em;
}
```



Figür 9-e-2

Şimdi de, butonumuzun etrafında bulunan tarayıcı standartı çerçevelerden kurtulmaya çalışalım:

```
form div input {
    color:#CCCCCC;
    background:#666666;
    font-weight:bold;
    font-size:0.9em;
    border:none;
}
```

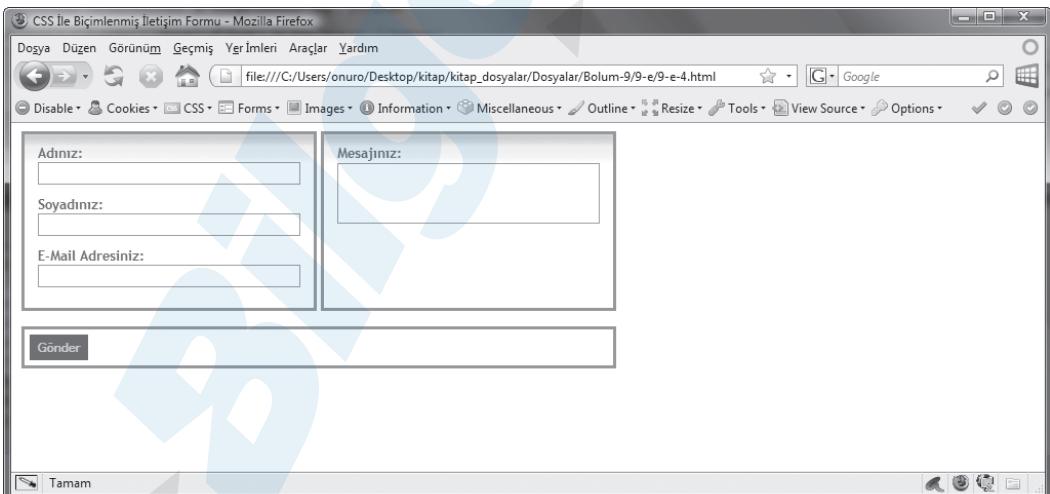


Figür 9-e-3

Çerçeveleri kaldırıldıktan sonra, metnimiz buton içerisinde biraz sıkışmış görünüyor.

Son olarak butonumuza iç boşluk uygulayarak buton görünümünü biraz daha ferah hale getirelim:

```
form div input {
color:#CCCCCC;
background:#666666;
font-weight:bold;
font-size:0.9em;
border:none;
padding:5px;
}
```



Figür 9-e-4

Hepsi bu kadar.

Böylece, CSS yardımıyla tarayıcı standardında oldukça sıkıntılı görünen formlarımızı nasıl daha albenili hale getirebileceğimizi görmüş olduk. Aklınızdaki tasarım ve uygulama fikirlerini CSS ile biçimlerken bol bol deneyerek kendinizi geliştirebilirsiniz.



10

CSS ile Veri Tablolarını Stillendirmek

10 css ile Veri Tablolarını Stillendirmek

- (x)HTML ile Erişilebilir Tablo Yapısı Hazırlamak
- Tablo ve İçerigini Stillendirmek
- Tablolarda Arkaplan Kullanmak

CSS ile Veri Tablolarını Stillendirmek

Kitabın ilk bölümünde hazırladığımız tasarımların CSS desteğiyle tablosuz olması gerekiğinden, geçerli nedenleri sıralayarak bahsetmiştim.

Buna rağmen tabloları gereken noktalarda kullanmamız gerekiğinden bahsetmiştim. Burada tasarım yapısı oluşturmak için değil, veri listeleme işlemlerinde ihtiyacımız olabileceğinin altını çizmek istiyorum.

Bir takvim ya da bir işyerindeki çalışanların bilgi tablosu gibi uygulamalarda `<table>` etiketi kullanmamız gerekecek.

Her ne kadar HTML tabloların biçimlenmesinde kendi etiket özelliklerine sahip olsa da, biz bu biçimlemeleri de CSS yardımıyla oluşturup stilsiz görünümde erişilebilir, stilli görünümde oldukça sık veri tabloları oluşturmaya çalışacağız.

Öncelikle, erişilebilir tabloların hazırlanışına bir giriş yapalım.

10.a (x)HTML ile Erişilebilir Tablo Yapısı Hazırlamak

Bir şirketin çalışanlarının eğitim ve yaş durumunun listelendiği bir doküman hazırlamak istedigimizde, bunu ziyaretçinin kolay anlayabileceği en pratik şekilde tablo elementlerini kullanarak hazırlayabiliriz.

Tablomuzu oluştururken, mümkün olduğunda açıklayıcı öğelere yer verecek, bilgisayar dışında farklı platform ve uygulamalardan erişilmesi durumunda anlaşılırlığını koruması için erişilebilir elementleri kullanacağız.

Öncelikle yeni bir HTML dokümanı oluşturalım ve tablo ana elementimiz olan `<table>` etiketini açalım:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title>CSS İle Biçimlenmiş İletişim Formu</title>
</head>
<body>
<table summary="X Şirketi Çalışan Bilgileri Tablosu">

</table>
</body>
</html>
```

Burada `table` etiketinin içine eklediğimiz `summary` (özet başlığı) özelliği ile tablomuzun konusu hakkında bilgiyi ziyaretçilerimizle paylaşmış oluyoruz. Bu tablonun özellikle stilsiz görünümünde içeriği hakkında bilgiyi daha iyi yansıtılmasına yardımcı olacaktır.

Şimdi, aynı başlığımızı tablo başlığımızın elementi (`caption`) olacak şekilde yerlestirelim.

```
<table summary="X Şirketi Çalışan Bilgileri Tablosu">

<caption>X Şirketi Çalışan Bilgileri Tablosu</caption>
</table>
```

Böylece tablomuzun içeriğiyle ilgili bilgileri tanımlamış olduk. Stillendirmemizi ağırlıklı olarak summary özelliği yerine caption elementine yapacağız. Zira tarayıcıda görüntülenen element caption olacaktır.

Hatırlarsanız evvelce form öğelerimizi biçimlerken onları mantıksal olarak gruplamak için field-set elementi içine almıştık. Erişilebilir tablolamadaysa benzeri işlevi gören ve tablonun içerik ve başlangıç bölümünü olduğunu ifade eden <thead> elementimiz mevcut.

Dolayısıyla, tablomuzun bilgi kategorilerini tanımlayan (bizim örneğimizde **çalışan adı, yaşı, eğitim durumu**) içeriğimizi thead ile kapsayıp içeriğin mantıksal olarak üst kısmının hazırlandığını belirleyeceğiz.

Kullanacağımız tablo kutu elementleriyle her zamanki gibi satırlar için <tr> etiketleri için sütunlarımız için klasik <td> kutuları yerine daha tanımlayıcı elementler olan table headers <th> öğelerini kullanacağımız.

Aslında th elementleri HTML'in eski zamanlarından beri mevcut idi, ancak CSS içi ile biçimlenmenin popüler hale gelişine kadar kullanışlı sayılmıyordu.

Bu kutu öğelerini kullanırken, kapsam (scope) özelliklerine sütun/kolon yapısı olduklarını tanımlayacak col değerini ekleyeceğiz.

Şimdi dilerseniz tüm bu bahsettiğim işlemleri tablomaza içerikleriyle beraber uygulayalım:

```
<table summary="X Şirketi Çalışan Bilgileri Tablosu">

<caption>X Şirketi Çalışan Bilgileri Tablosu</caption>
<thead>
<tr>
<th scope="col">Çalışan Adı</th>
<th scope="col">Yaşı</th>
<th scope="col">Eğitim Durumu</th>
</tr>
</thead>
</table>
```

Tablomuzun bilgi kategorilerini yerleştirmiş olduk.

Şimdi tablomuzun içeriğini satır ve standart sütun (td) elementlerimizle, tablo içeriği olduğunu tanımlayan <tbody> elementimizle kapsayarak girebiliriz:

```
<table summary="X Şirketi Çalışan Bilgileri Tablosu">

<caption>X Şirketi Çalışan Bilgileri Tablosu</caption>
<thead>
<tr>
<th scope="col">Çalışan Adı</th>
<th scope="col">Yaşı</th>
```

```

<th scope="col">Eğitim Durumu</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="#">Erkan Kocatürk</td>
<td>23</td>
<td>Lise</td>
</tr>
<tr>
<td><a href="#">Hatice Ersöz</td>
<td>27</td>
<td>Üniversite</td>
</tr>
<tr>
<td><a href="#">Emin Çetin</td>
<td>34</td>
<td>Üniversite</td>
</tr>
</tbody>
</table>

```

Tablomuz neredeyse hazır.

Tablo yapımızın üst (thead) ve içerik (tbody) bölümlerini tamamlamışken, dilerseniz toplam çalışan sayısını da tablonun kapanış bölümü (tfoot) olarak tanımladığımız bir bölüme yazdırılim:

```

<table summary="X Şirketi Çalışan Bilgileri Tablosu">

<caption>X Şirketi Çalışan Bilgileri Tablosu</caption>
<thead>
<tr>
<th scope="col">Çalışan Adı</th>
<th scope="col">Yaşı</th>
<th scope="col">Eğitim Durumu</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="#">Erkan Kocatürk</td>
<td>23</td>
<td>Lise</td>
</tr>
<tr>
<td><a href="#">Hatice Ersöz</td>

```

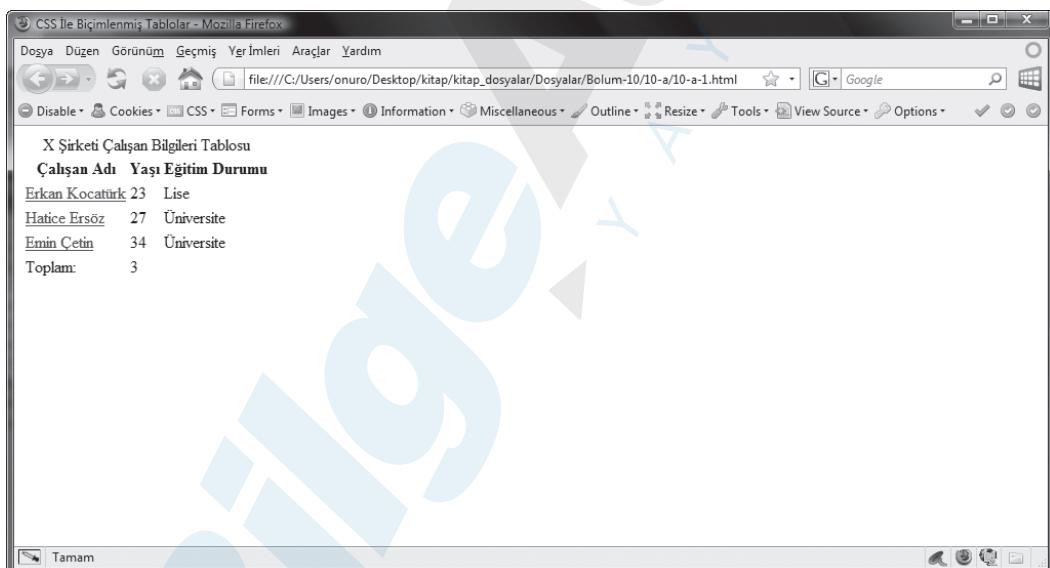
```

<td>27</td>
<td>Üniversite</td>
</tr>
<tr>
<td><a href="#">Emin Çetin</td>
<td>34</td>
<td>Üniversite</td>
</tr>
</tbody>

<tfoot>
<tr>
<td> Toplam: 3</td>
<td colspan="3">&ampnbsp;</td>
</tr>
</tfoot>
</table>

```

Hazırlamış olduğumuz tablomuza bir göz atalım:



Figuır 10-a-1

Stilsiz olarak, içeriği düzgün bir şekilde listelenen erişilebilir bir veri tablosu oluşturmuş olduk.

Bundan sonraki aşamalarda, bu tablonun görünümü üzerinde radikal değişimlere varan stillendirme uygulayarak veri tablomuzu oldukça estetik hale getireceğiz.

10.b Tablo ve İçerigini Stillendirmek

Hazırlamış olduğumuz tablomuz, şu anki stilsiz haliyle pek okunaklı ve albenili görünmüyordu.

CSS yardımıyla, tablomuzun temel görünüm özelliklerini biçimleyerek biraz daha stil sahibi görüne kavuşması için çalışacağız.

Tablomuzdan önce, ben stil dosyasında doküman geneli ve link özelliklerini tanımlıyorum.

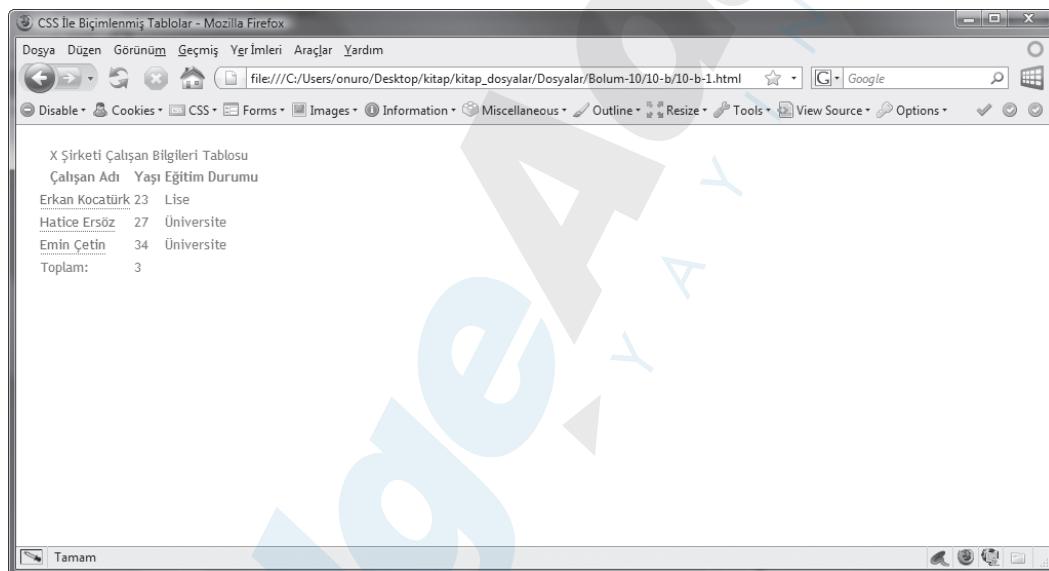
```

body {
    font:9.5pt "Trebuchet MS";
    margin:20px;
    color:#777;
}

a {
    color:#666;
    text-decoration:none;
    border-bottom:1px dotted #666;
}

a:hover {
    color:#ac0000;
    border-bottom-color:#ac0000;
}

```



Figür 10-b-1

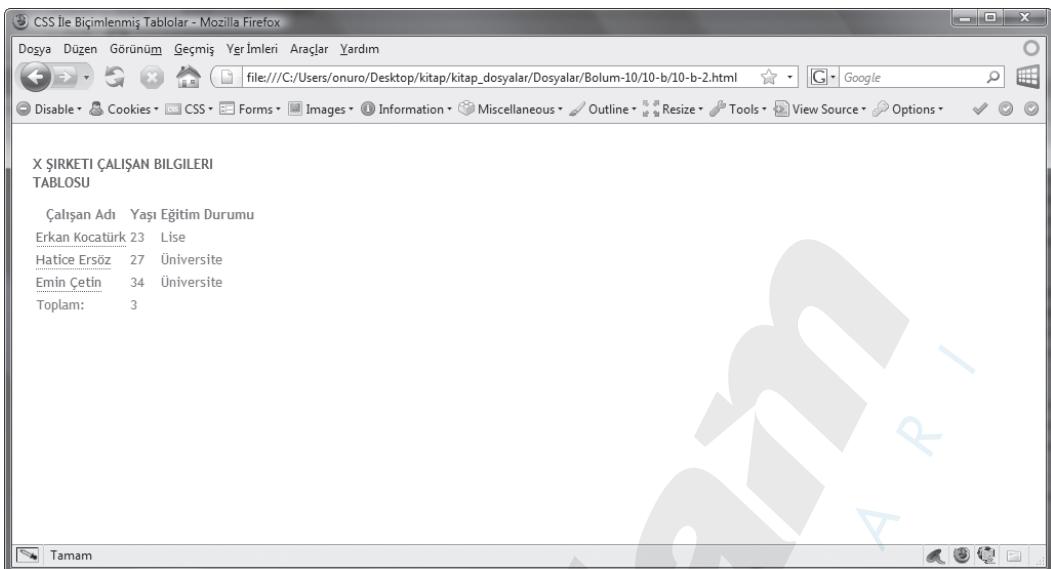
Doküman ve metnimiz daha okunaklı bir yapıya kavuştu ve linklerimizin atlındaki çizgi kalkarak yerini noktalı alt çerçeve alarak hover durumunda link metin rengi vişne kırmızısı olacak şekilde değişti.

Sırayla gidelim, ilk olarak tablomuzun içindeki tablo başlığını (*caption*) biçimlendirerek onu şu anki silik görünümünden kurtaralım:

```

caption {
    padding:10px 0;
    font-size:105%;
    font-weight:bold;
    text-align:left;
    text-transform:uppercase;
    color:#ac0000;
}

```

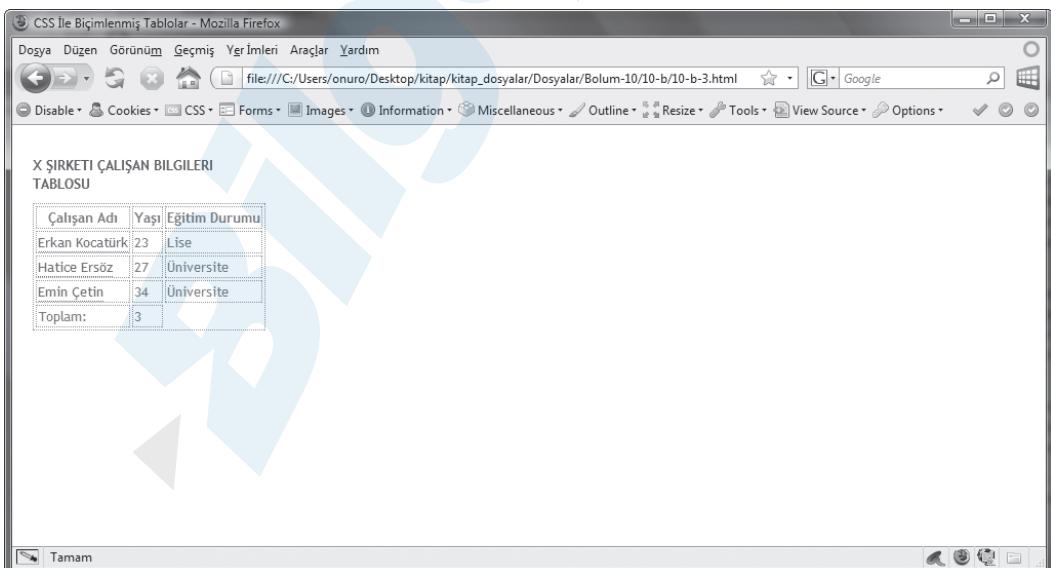
**Figür 10-b-2**

Tarayıcı standardında ortalanmış olan açıklama başlığımız, tanımladığımız stillerle biraz daha büyük punto boyutuna sahip olarak vişne rengi, kalın, sola yaslı ve büyük harflerle göرündülenecek bir yapıya kavuştu.

Verdiğimiz 10 piksellik (sol taraf hariç olmak üzere üst, sağ ve alt taraf için) iç boşlukla da daha ferah bir yapıda görünüyor.

Şimdi tablomuzun çerçeve stilleri üzerinde çalışabiliriz. Tablomuza, kendisi dahil olmak üzere tüm ana kutu yapısı elementlerine 1 piksellik noktalı gri birer çerçeve atayalım:

```
table, td, th {  
border:1px dotted #999;  
}
```

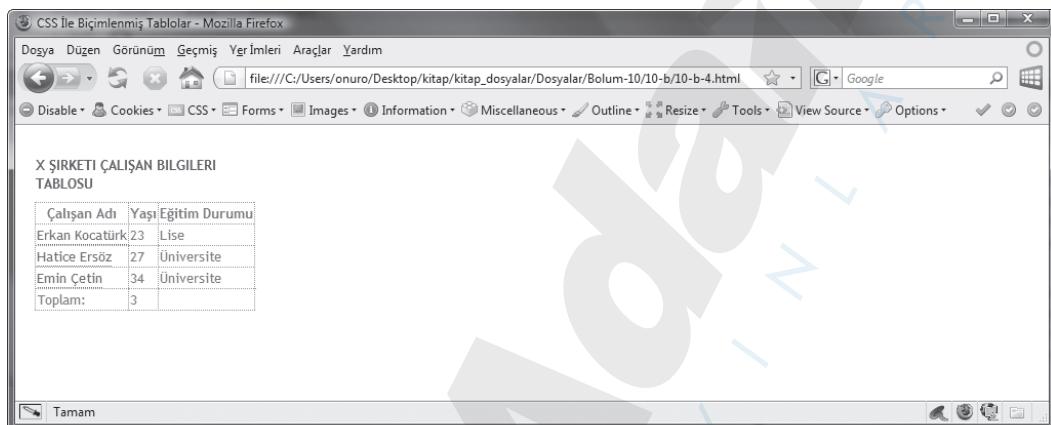
**Figür 10-b-3**

Uyguladığımız işlem, tablomuzun sık çerçevelere sahip olmasını sağladı.

Ancak tablo ve içindeki kutuların sahip olduğu çerçeveler arasındaki boşluk göze çarpıyor. CSS yardımıyla tarayıcıya tablodaki elementlerin çerçevelerini tek bir çerçeve ögesi gibi görüntületebiliriz.

Tablomuzda bunu yapmamızı sağlamak için bu amaçla kullanılan border-collapse özelliğine collapse değeri atayacağız:

```
table, td, th {
border:1px dotted #999;
border-collapse:collapse;
}
```

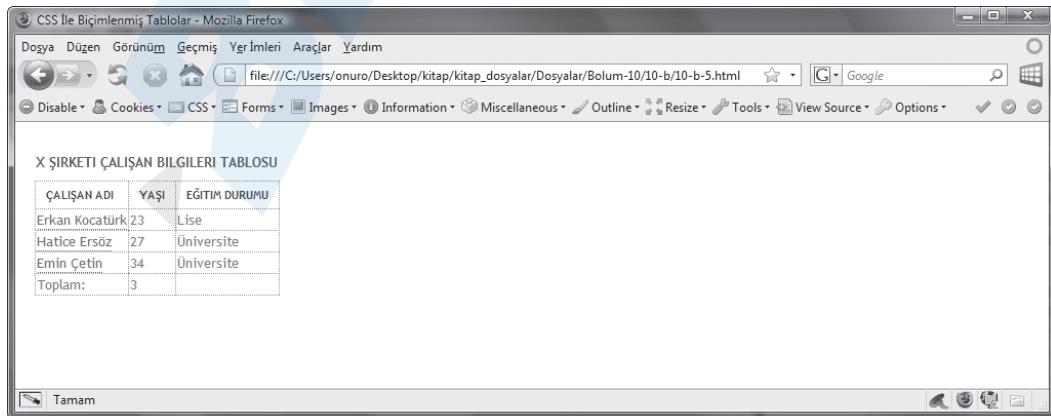


Figür 10-b-4

Artık hücrelerimizle tablomuzun çerçeveleri bitiştii.

Şimdi dilerseniz table header içeriğimizi, yani veri tablomuzun üstteki kategori başlıklarını biçimleyelim:

```
th {
padding:5px 10px;
font-size:90%;
text-transform:uppercase;
text-align:left;
color:#333;
}
```

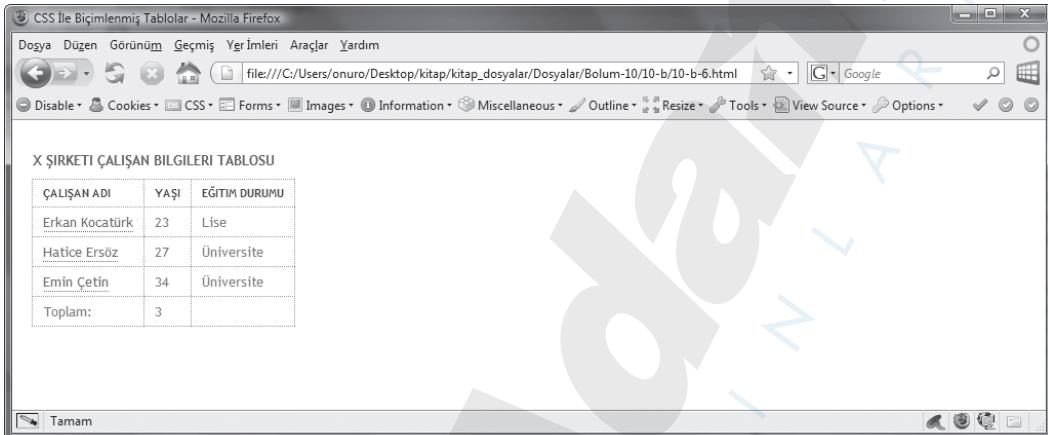


Figür 10-b-5

Biraz daha belirginleşen tablo kategori başlıklarımız, gri kalın ve büyük harflere dönüşerek sola yaslandılar, biraz iç boşluk aldılar ve yazıtipi boyutları üst element yazıtipi oranlarına göre %10 küçüldü.

Şimdi sıra kolon içeriğimizi biçimlemeye geldi. Veri içeriğimizin listelendiği kolonlarımızı iç boşluk uygulayarak biraz ferahlatalım:

```
td {  
padding:5px 10px;  
}
```

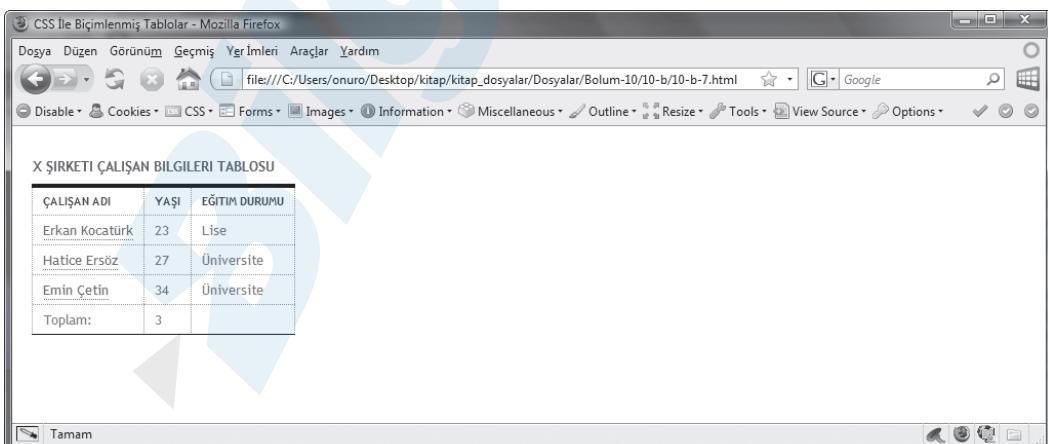


Figür 10-b-6

İlk haline oranla çok daha sık bir tablo görünümü elde etmiş olduk.

Son olarak, tablonun başı ve sonunu belirginleştirecek biçimlemeyi tablomuzun üst ve alt çerçevesini stillendirerek yapalım:

```
table {  
border-top:4px solid #000;  
border-bottom:1px solid #000;  
}
```



Figür 10-b-7

İşte bu kadar. Oldukça estetik ve erişilebilir bir tablo elde ettik.

Bir sonraki aşamamızda, tablomuzun arkaplan renkleriyle oynayarak ve grafik arkaplanlar atayaarak albenisini artırmaya çalışacağız.

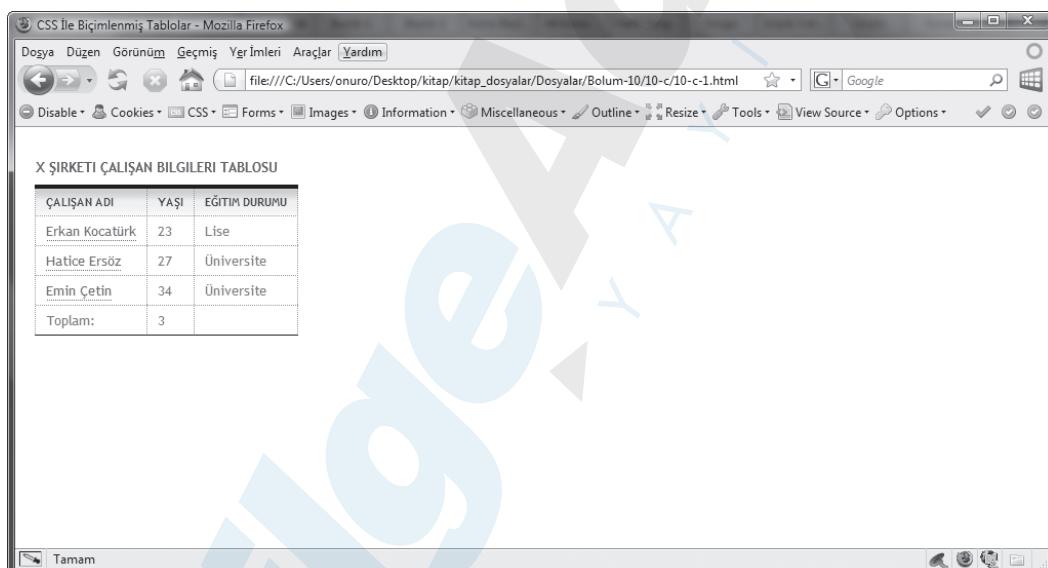
10.c Tablolarda Arkaplan Kullanmak

Arkaplan (background) özelliğini şu ana kadar birçok yerde, birçok element için sıkça kullanarak dokümanlarımızdaki elementlerin görünümüne en büyük görsel etkiye sağlamıştık.

Bu kez arkaplan stillendirmesini tablomuz ve içindeki öğeler için yapacağız.

Dilerseniz işe öncelikle tablo header'larımıza arkaplan grafiği atayarak başlayalım. `fieldset` örneğinde kullanmış olduğumuz grafiğin aynısını “`th_bg.jpg`” adıyla `th` arkaplanı olarak yatay tekrar edecek şekilde atayalım:

```
th {
background:url(th_bg.jpg) repeat-x;
padding:5px 10px;
font-size:90%;
text-transform:uppercase;
text-align:left;
color:#333;
}
```



Figür 10-c-1

Şimdi, tablo footer elementimiz olan `tfoot` için arkaplan rengini siyah olarak belirleyelim. Ancak metin rengimiz de siyaha yakın bir gri tonu olduğu için, bu kez metinlerimizde okunma problemi olacak.

Dolayısıyla `tfoot` için arkaplan rengini siyah uygularken, metni de beyaz renkle görüntüleyeceğiz:

```
tfoot {
background:#000;
color:#fff;
}
```

ÇALIŞAN ADI	YAŞI	EĞİTİM DURUMU
Erkan Kocatürk	23	Lise
Hatice Ersöz	27	Üniversite
Emin Çetin	34	Üniversite
Toplam:	3	

Figür 10-c-2

Bitti bölümünü oldukça belirginleştirdik.

Ancak tabloya atadığımız siyah alt çerçeveye, tfoot arkaplanı uyguladığımızdan artık bizim için estetik olmaktan çıkyor.

Dolayısıyla tablo alt çerçevesini (border-bottom) stillendiren satırı kaldırıyoruz:

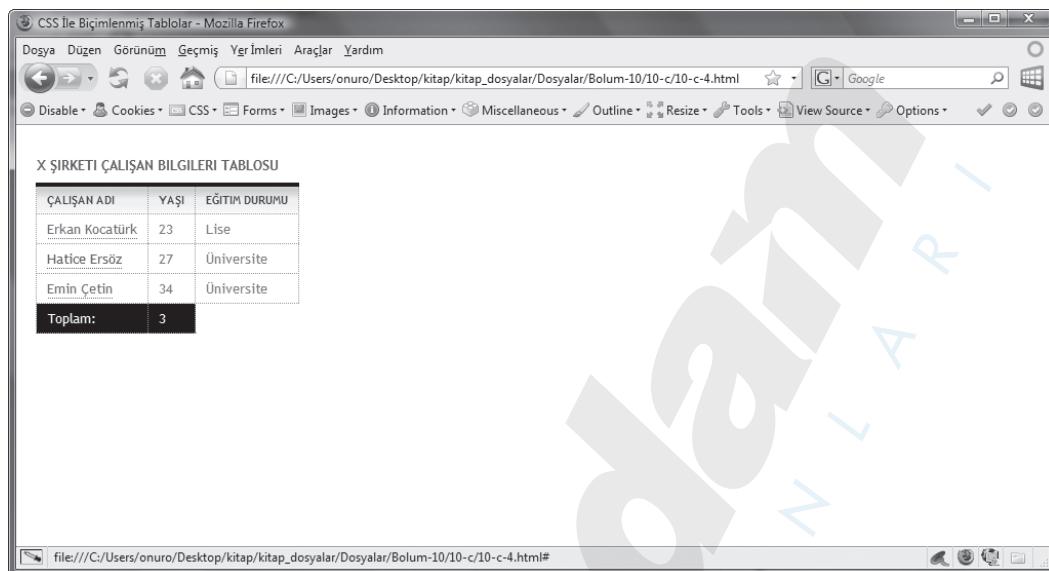
ÇALIŞAN ADI	YAŞI	EĞİTİM DURUMU
Erkan Kocatürk	23	Lise
Hatice Ersöz	27	Üniversite
Emin Çetin	34	Üniversite
Toplam:	3	

Figür 10-c-3

Nispeten daha az gözü yoran bir alt kısma sahip olduk, fakat tfoot elementinin devamındaki çerçevelerle ilgili hala sorunumuz var.

Amacımız, tfoot dışında kalan ve üst tanımlamadan (table, td, th) kalıtsallık alarak devam eden noktalı çerçeveleri kaldırmak. Bunun için, table elementimizde yalnızca üst çerçeve satırı kalmışken bir üstüne tüm çerçevelerin kaldırıldığı ibaresini yerleştiriyoruz. Bu, table elementimiz içinde net olarak yalnızca üst çerçevenin kalmasını sağlayacak:

```
table {
border:none;
border-top:4px solid #000;
}
```

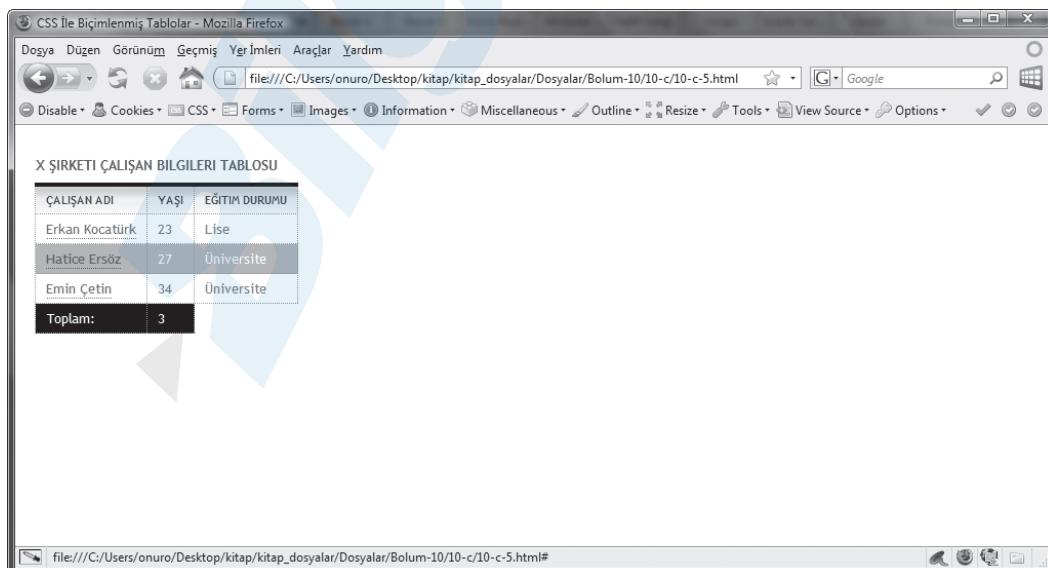


Figür 10-c-4

İlk satırda tüm çerçeveleri kaldırılan tablo, sonrasında gelen satırda siyah 3 piksellik çerçeve tanıtılmış halen bulunduğu için 3 taraf (sol, sağ, alt) hariç tepede çerçevesini korumaya devam ediyor.

Şimdi son olarak, tablomuzdaki `tbody` içeriğine hover uygulandığında arkaplan ve metin rengini değiştirerek hover durumunda tablo içeriğimize son derece çekici bir stil kazandıralım:

```
tbody tr:hover {
background:#b9b9b9;
color:#fff;
}
```



Figür 10-c-5

Göründüğü gibi, CSS kullanarak veri tablolarımızı da son derece çekici hale getirebiliyoruz.

Bu konuya birlikte, kitabın CSS temel hiyerarşik yapı, konum ve içerik stillendirmeyle ilgili son bölümünü de görmüş olduk.

Buraya kadar yaptığımız çalışmalarda, pek komplike ve çokça satır içeren gelişmiş CSS uygulamaları yapmamış olmamamıza rağmen, ufak stillendirme ve biçimlemelerle dahi Web dokümanlarımızın görünümünde sınırsız varyasyonlar yapabildiğimizi anlatmaya çalıştım.

Bundan sonraki bölümde, Photoshop ile tasarlanmış bir sayfayı ele alıp, onu CSS ile adım adım hayata geçirmeye çalışacağız.



11

**CSS ile Bir
Tasarımı Hayata
Geçirmek**

11 CSS ile Bir Tasarımı Hayata Geçirmek

- Tasarımın CSS Uygulamasını Planlamak:
Tasarım Öğelerinin Element ve Stil
Karşılıklarını Organize Etmek
- Dokümanı Stilsiz Olarak Hazırlamak:
- Üst Kısmı Stillendirmek:
- Orta Kısmı Stillendirmek:
- İçerik Kısmını Stillendirmek
- Alt Kısmı Stillendirmek:

CSS ile Bir Tasarımı Hayata Geçirmek

Bu bölüme kadar gördüklerimiz, dokümanlarımızdaki element ve grup öğelerin estetik bir biçimde stillendirilmeleriyle ilgiliydi.

Bu bölümdeyse, Photoshop ile tasarlamış olduğumuz bir Web dokümanını tüm öğelerini ele alarak (x)HTML ve CSS çıktısıyla hayata geçirmeye çalışacağız.

Web tasarımları, nasıl ki üretimi esnasında planlama ve organizasyon gerektiriyorsa, tasarımın Web dokümanı haline getirilerek görsel halinin minimum farklarla hata geçebilmesi için de planlanması gereklidir.

Örnek olarak ele alacağımız çalışma, Apple iPod üzerine hazırlanan tek sayfalık görsel bir tanıtım dokümanı olacak:

The screenshot shows the official product page for the iPhone 3G. At the top left is the Apple logo and the text "Apple iPhone 3G". Below it is a large image of the iPhone 3G device. To the right of the phone is a vertical column of text: "IPHONE 3G - APPLE INCORPORATED". In the center, there's a heading "Mobil İletişimin Yeni Gözdesi Parmaklarınızın Ucunda!" followed by the subtext "3G versiyonu ile daha hafif, daha ucuz." Below this, there's a paragraph about the iPhone's features and a quote: "iPhone, bir cep telefonunun neler yapabileceğini yeniden tanımlıyor." At the bottom, there are three sections: "Özellikler" (Features) with icons for E-posta, Safari, iPod, YouTube, SMS, GPS'li Haritalar, and a "Detaylar" link; "Kutu İçindekiler" (What's in the box) listing 3G iPhone, Mikrofonlu Stereo, Kulaklık, USB Kablo Konnektörü, USB Adaptör, and Kullanım Kilavuzu, with a "Detaylar" link; and "Güç ve Batarya" (Power and Battery) detailing a 3.7V 1420 mAh Li-Ion battery, a 10-hour talk time, and a 5-day stand-by time, with a "Detaylar" link.

Copyright 2008 Apple INC.

Görsüntü 11-1

Photoshop ile hazırlanan bu dokümanı, mümkün olduğunda bu gördüğümüz biçimde Web dokümanı haline getirmeye çalışacağız.

11.a Tasarımın CSS Uygulamasını Planlamak: Tasarım Öğelerinin Element ve Stil Karşılıklarını Organize Etmek

Tasarımımıza genel olarak baktığımızda, çalıştığımızın görsel ağırlıklı olduğunu görürüz:



Görsüntü 11-a-1

Ancak bu, görsel öğeleri doğrudan dokümanımıza yerleştirebileceğimiz anlamına gelmiyor.

Ziyaretçilerimiz, CSS yüklü, stilli görünümde dokümanımızın bu sık halini görüntülerken; stilsiz gezen veya farklı bir platformdan erişen (cep telefonu, kiosk, e-mail arayüzü, el bilgisayarı vb.) ziyaretçilerimiz için de içeriğin erişilebilir ve gezilebilir olması gereklidir.

Bu durumda; dokümanımızı öncelikle stilsiz olarak oluşturup, daha sonra adım adım, bazı bölümlerde **“Görsel Yerdeğşim”** ve benzeri teknikleri uygulayarak stillendireceğiz ve sonuç olarak stilli görünümde oldukça sık, stilsiz görünümde de kolaylıkla gezilebilen anlaşılır bir Web dokümanı elde edeceğiz.

Birimleme öncesi, dokümanı öncelikle hayali kutulara böleceğiz ve onlara id selektör atamalarını yaparak içeriği gireceğiz. Bunun mantığını şu şekilde kuralım:

1. Tasarımımıza bakacak olursak, ortalanmış bir sayfa düzenine ihtiyacımız olacak. Bunu bize, `#anakutu` div kutumuz sağlayabilir.
2. Tüm öğeleri kapsayarak ortalanacak bu anakutumuz içinde aşağıdaki alt ana kutular buluncak
 - a. `ust_kutu`
 - b. `orta_kutu`
 - c. `icerik_kutusu`
 - d. `alt_kutu`

Dolayısıyla tahmin edeceğiniz üzere, dokümanımızı aynı normal bir yazı dokümanı gibi giriş, gelişme ve sonuç aşamalarına böliyoruz. Bu, hem doküman, hem de görsel hiyerarşisini oluşturacak.

3. Kutularımızın içindeki elementleri oluşturup içeriği gireceğiz.

Bu bahsettiğim adımlardan ikincisi olan hiyerarşik yapıdaki düzene Photoshop üzerinde göz atalım:

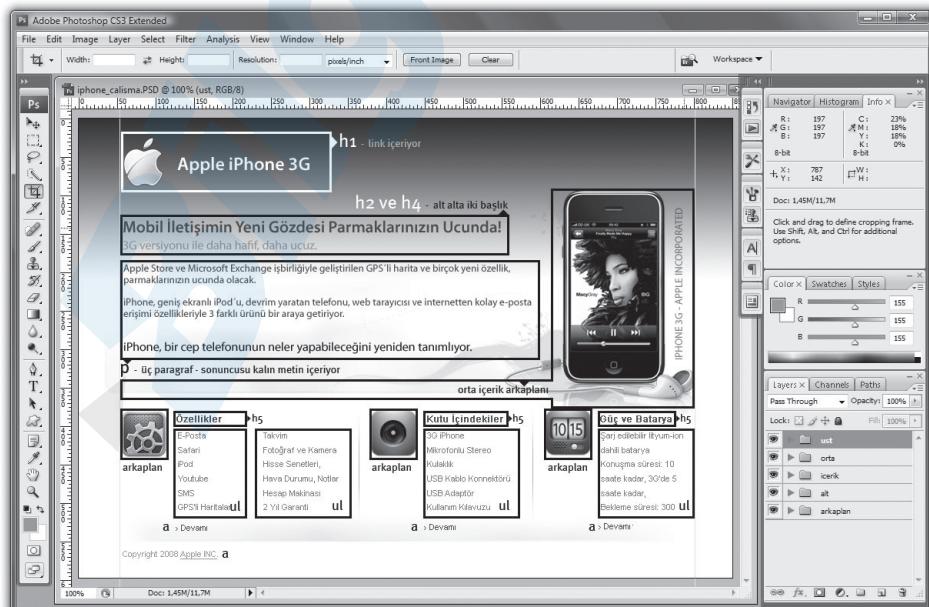


Görüntü 11-a-2

Toplamı anakutu içerisinde yer alacak bu id selektörlü div kutuları, dokümanınızın yapısını oluşturup öğeleri bir arada tutan ana elementler olacaklar.

Ben id selektör isimlendirmelerini yaparken anlaşılması kolay olması için şu ana kadar hep Türkçe karşılıklar kullandım. Bununla birlikte, **Görüntü 11-a-2**'de gördüğümüz selektör adlarının yanındaki kavramlar, evrensel olarak kullanılan tanımlardır.

Tasarımımız çoğunlukla görsel öğeler içerdiginden özellikle üst ve orta kısımlarda “görsel yerdeğşim” yapmamız gerekecek. Dolayısıyla dokümanda grafik olarak görünse de her elementin birer karşılığı olacak:



Görüntü 11-a-3

Dokümanımızı oluştururken, kutularımızı hazırlayarak içeriğimizi görsel öğelerimizin ve metinlerimizin element karşılığını girerek hazırlayacağız.

Bir sonraki adımda, ilk aşamamız olan dokümanımızı stilsiz olarak hazırlamakla başlayacağız.

11.b Dokümanı Stilsiz Olarak Hazırlamak

Bir önceki bölümde doküman hiyerarşisini ve element karşılıklarını nasıl hazırlamamız gerekiğine dair bazı fikirler edindiğimize göre, sayfamızın stilsiz halini hazırlama işlemeye geçebiliriz.

Öncelikle, yapısal öğelerimizi, yani kutu elementlerimizi hazırlayacağız. Ben kutu yoğunluğu CSS ile biçimlerken akılda soru işaretini bırakmaması için, hiyerarşik etiket kapanışlarını yorum satırıyla belirtiyor olacağım. İlk olarak anakutumuz ile başlayalım:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title>iPhone 3G</title>
</head>

<body>
<div id="anakutu">

</div><!-- anakutu burada kapanıyor -->

</body>
</html>
```

Anakutumuz tüm diğer kutularımızı kapsayıp, dokümana ortalanarak yerleşen bir taşıyıcı görevi görecektir.

İçine girecek öğelerden ilkiyse içinde bir link olan ana başlığımızın yer aldığı **ust_kutu** div elementimiz:

```
<div id="anakutu">

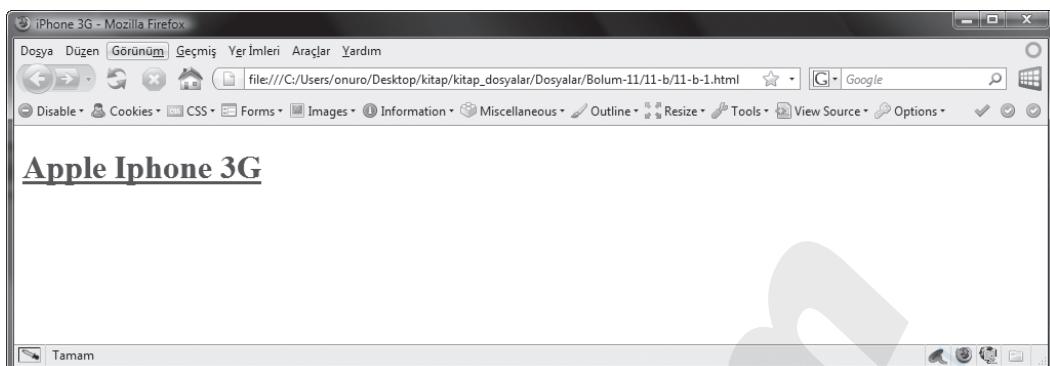
<div id="ust_kutu">

<h1><a href="#">Apple Iphone 3G</a></h1>

</div><!-- #ust_kutu burada kapanıyor -->

</div><!-- #anakutu burada kapanıyor -->
```

Buraya kadar hazırladığımız dokümanımıza bir göz atalım:



Görüntü 11-b-1

Div elementleri stilsiz durumdayken görsel bir değişim arz etmediği için şu an itibarıyle anakutu ve üst kutumuzu tarayıcıda belirgin olarak göremiyoruz, ancak elbette bunun bir önemi yok.

Üst kutu ve içeriğimizle işimiz bittiğine göre, orta kutumuzu hazırlayalım.

Tasarıma baktığımızda, bu kısmın tamamen grafik yapısında olduğunu görüyoruz. Stilsiz görünümde de bu içeriği verebilmemiz için, daha evvel element karşılığını belirlediğimiz öğeleri yine de yerleştirmemiz gerekecek.

Ziyaretçi bu sayfaya stilsiz görüntülemeyle girdiğinde, iPhone gibi tanıtımı yapılrken görselliğe önem verilen bir ürün için telefonun fotoğrafını koymak mantıklı olacaktır. Stilli görünümde her ne kadar telefon görselimizin bulunduğu arkaplan grafiği yer alacaksa da, stilsiz görünümde de bu `` görselini kullanacağız.

Şimdi, orta kutu div elementimizi oluşturalım ve içeriğimizi girelim:

```

<div id="orta_kutu">

  <h2>Mobil İletişimin Yeni Gözdesi Parmaklarınızın Ucunda!</h2>
  <h4>3G versiyonu ile daha hafif, daha ucuz.</h4>

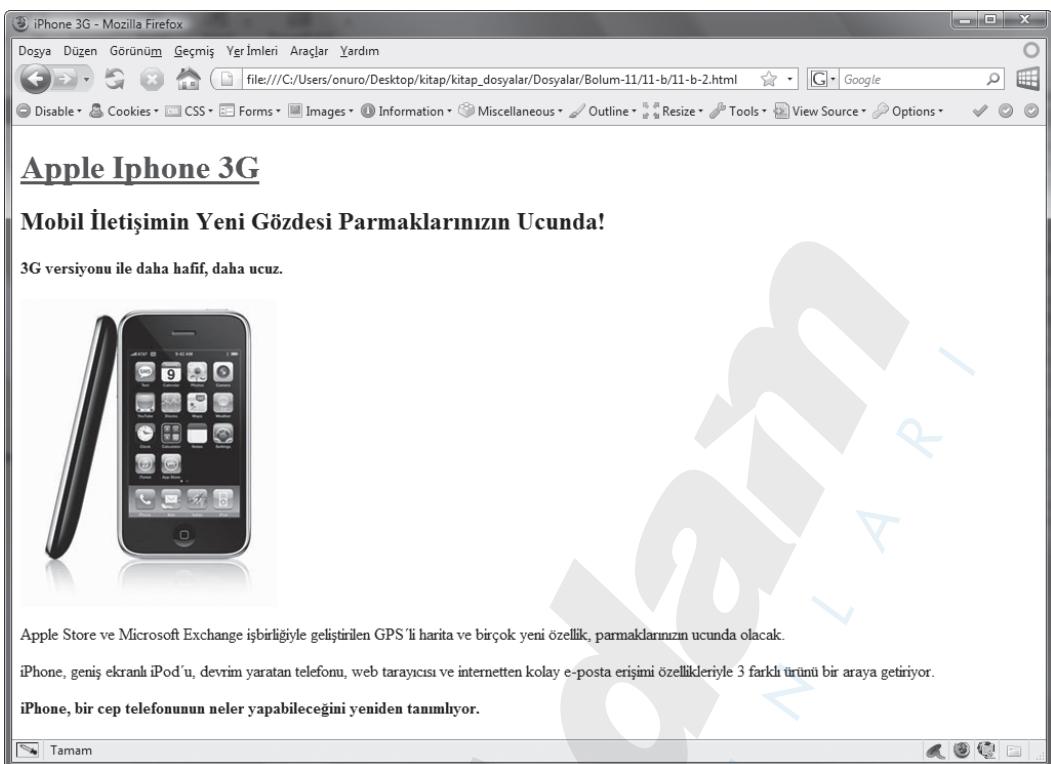
  <p>Apple Store ve Microsoft Exchange işbirliğiyle geliştirilen
  GPS'li harita ve birçok yeni özellik, parmaklarınızın ucunda
  olacak.</p>

  <p>iPhone, geniş ekranlı iPod'u, devrim yaratan telefonu, web
  tarayıcısı ve internetten kolay e-posta erişimi özellikleyle 3
  farklı ürünü bir araya getiriyor.</p>

  <p><strong>iPhone, bir cep telefonunun neler yapabileceğini
  yeniden tanımlıyor.</strong></p>

</div><!-- #orta_kutu burada kapanıyor -->

```

*Figür 11-b-2*

Üst ve orta bölüm kutularımız ve içeriğimiz tamamlandı.

Şimdi içerik kutumuz ve onun içeriğini hazırlamamız gerekiyor. Burada fark ettiyseniz toplamda 3 ana liste öğesi (`ul`) ve onları ayıran bazı grafik çizgileri var:

Özellikler	Kutu İçindekiler	Güç ve Batarya
E-Posta	Takvim	Şarj edilebilir lityum-ion dahili batarya
Safari	Fotoğraf ve Kamera	Konuşma süresi: 10 saatte kadar, 3G'de 5 saatte kadar,
iPod	Hisse Senetleri,	Bekleme süresi: 300
Youtube	Hava Durumu, Notlar	
SMS	Hesap Makinası	
GPS'li Haritalar	2 Yıl Garanti	
> Devamı	> Devamı	> Devamı

Görüntü 11-b-1

Dolayısıyla bu öğeleri de kolayca stillendirebilmemiz için id selektör kutularına bölmemiz gerekiyor. Liste kutularına başlıklarındaki adları vermek istersek, div elementlerine atayacağımız id selektör adları aşağıdaki gibi olabilir:

1. ozellikler
2. kutu_icindekiler
3. guc_ve_batarya

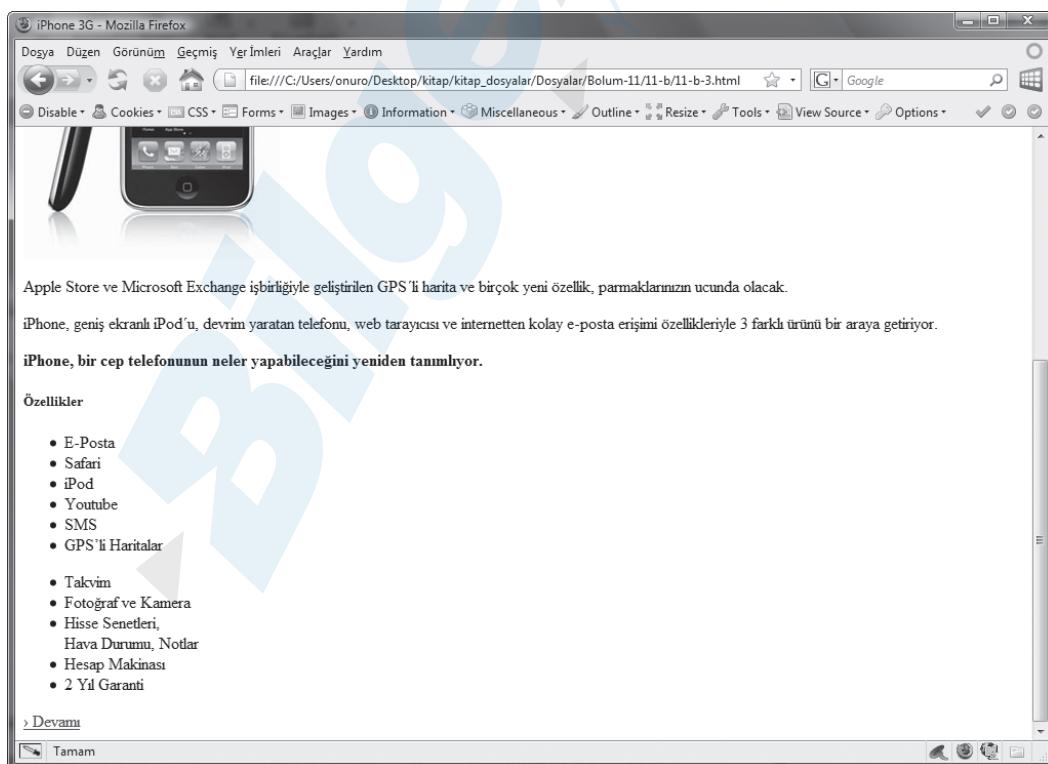
Şimdi dilerseniz öncelikle içerik kutumuzu oluşturup içine ilk olarak “ozellikler” kutusunu ve içeriğini girelim:

```

<div id="icerik_kutusu">
    <div id="ozellikler">
        <h5>Özellikler</h5>
        <ul>
            <li>E-Posta</li>
            <li> Safari</li>
            <li> iPod</li>
            <li> Youtube</li>
            <li> SMS</li>
            <li> GPS'li Haritalar</li>
        </ul>

        <ul>
            <li>Takvim</li>
            <li> Fotoğraf ve Kamera</li>
            <li> Hisse Senetleri, <br />
                Hava Durumu, Notlar</li>
            <li> Hesap Makinası</li>
            <li> 2 Yıl Garanti </li>
        </ul>
        <div class="link"><a href="#">, Devamı</a></div>
    </div><!-- #ozellikler burada kapanıyor -->
</div><!-- #icerik_kutusu burada kapanıyor -->

```



Figür 11-b-3

Aynı işlemi “kutu içindekiler” ve “guc_ve_batarya” kutularımız için de uygulayacak olursak:

```
<div id="icerik_kutusu">

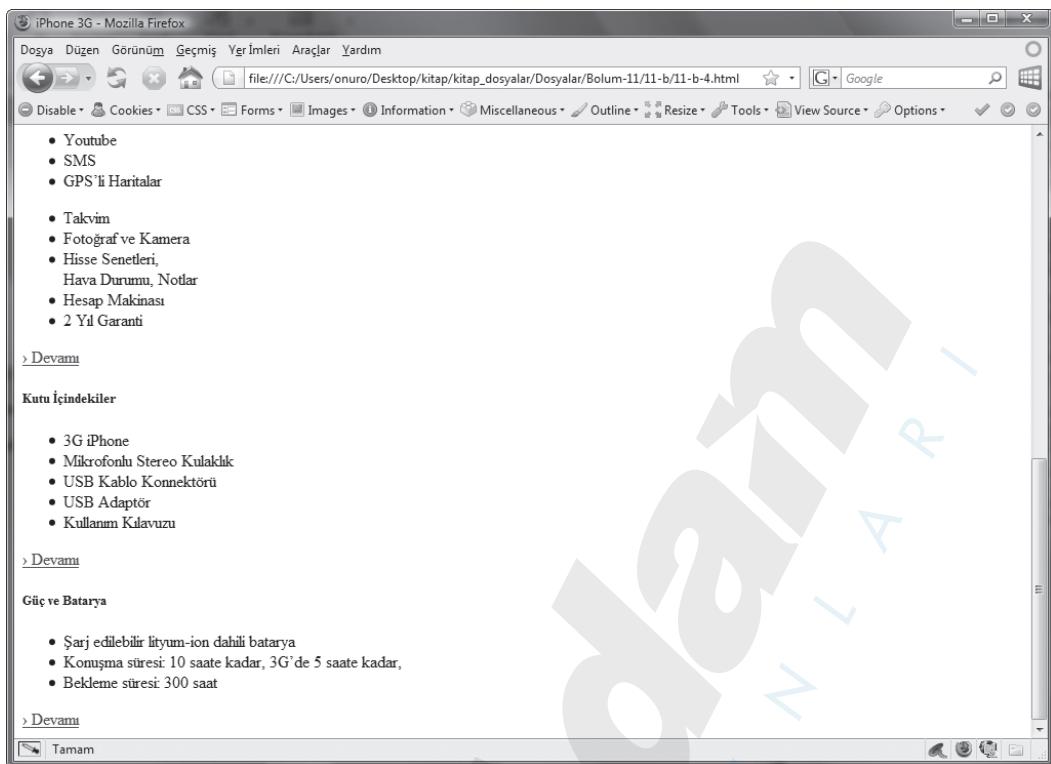
    <div id="ozellikler">
        <h5>Özellikler</h5>
        <ul>
            <li>E-Posta</li>
            <li> Safari</li>
            <li> iPod</li>
            <li> Youtube</li>
            <li> SMS</li>
            <li> GPS'li Haritalar</li>
        </ul>

        <ul>
            <li>Takvim</li>
            <li> Fotoğraf ve Kamera</li>
            <li> Hisse Senetleri, <br />
                Hava Durumu, Notlar</li>
            <li> Hesap Makinası</li>
            <li> 2 Yıl Garanti </li>
        </ul>

        <div class="link"><a href="#">, Devamı</a></div>
    </div><!-- #ozellikler burada kapanıyor -->

    <div id="kutu_icindekiler">
        <h5>Kutu İçindekiler</h5>
        <ul>
            <li>3G iPhone</li>
            <li>Mikrofonlu Stereo Kulaklık</li>
            <li>USB Kablo Konnektörü</li>
            <li>USB Adaptör</li>
            <li>Kullanım Kılavuzu</li>
        </ul>
        <div class="link"><a href="#">, Devamı</a></div>
    </div><!-- #kutu_icindekiler burada kapanıyor -->

    <div id="guc_ve_batarya">
        <h5>Güç ve Batarya</h5>
        <ul>
            <li>Şarj edilebilir lityum-ion dahili batarya</li>
            <li> Konuşma süresi: 10 saat kadar, 3G'de 5 saat kadar</li>
            <li> Bekleme süresi: 300 saat</li>
        </ul>
        <div class="link"><a href="#">, Devamı</a></div>
    </div><!-- #guc_ve_batarya burada kapanıyor -->
</div><!-- #icerik_kutusu burada kapanıyor -->
```

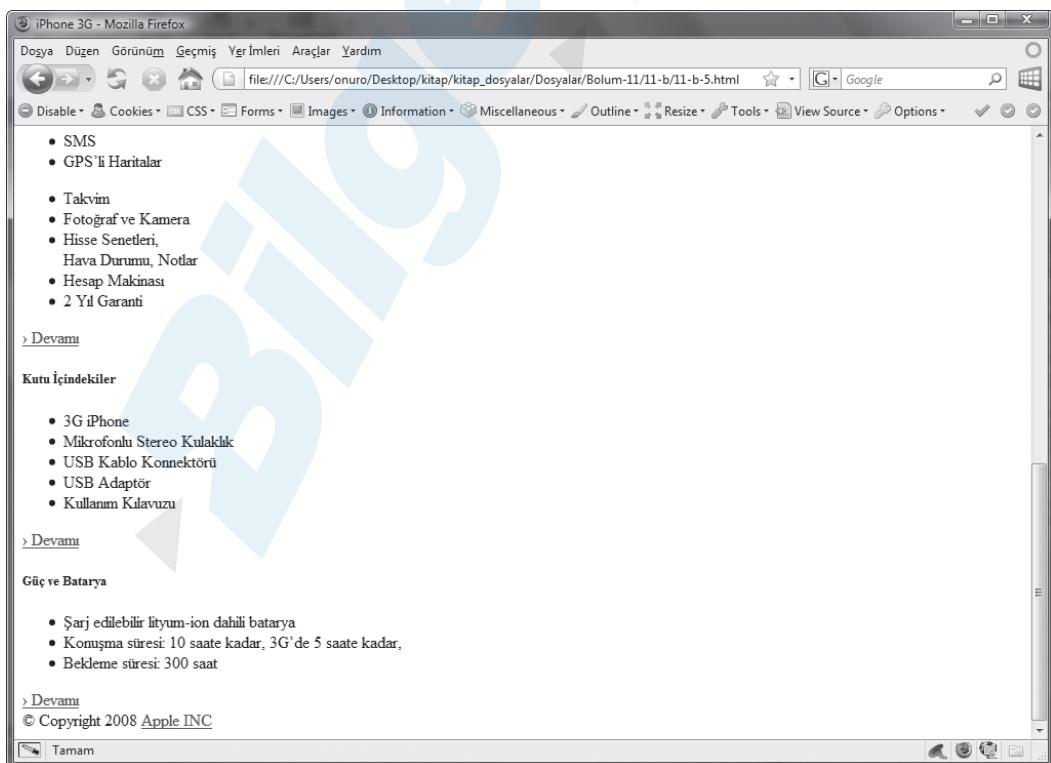


Figür 11-b-4

Dokümanımız neredeyse hazır.

Son olarak, alt kutu öğemizi oluşturup içeriğini girelim:

```
<div id="alt_kutu">&copy; Copyright 2008 <a href="#">Apple INC</a></div>
```



Figür 11-b-5

Stilsiz görünümde oldukça açıklayıcı ve işlevini getiren bir yapıda oluşturduğumuz HTML dokümanımızı, bir sonraki adımda anakutu ve üst kutu yapısını stillendirerek görselleştirmeye başlayacağız.

11.c Üst Kısmı Stillendirmek

HTML dokümanımız hazır olduğuna göre, hiyerarşik yapıyı baz alarak sırayla bölümlerini biçimlemeye başlayacağız.

Öncelikle tasarım görünümüne baktığımızda, dokümanımızın genişliği boyunca uzayan bir arkaplana sahip olduğunu, ve yapıların bu arkaplan üzerine oturması gerektiğini görürüz.

Dolayısıyla Photoshop çalışmamızda arkaplan grafiğini oluşturacak kısmı keserek uygun formatta kaydettikten sonra doküman geneli özelliklerimize yatay tekrar edecek şekilde atamamız gerekecektir. Ben grafiği, "arkaplan.jpg" adıyla kaydediyorum:



Görüntü 11-c-1

Doküman geneli özellikleriyle başlayacağımız için, anakutu div özelliklerini biçimlemeden önce body elementimiz için arkaplan atamalarını ve boşluk ayarlarını yapacağız.

Şu ana kadar kullandığımız CSS biçimleme yöntemi, tek sayfalık dokümanlarda sıkça başvurulan yöntem olan dahili stillendirmeydi. Bu örneğimiz de tek sayfalık olsa da; ben çoklu sayfalarda ortak görünüm ve merkezi kontrol için emsal teşkil etmesi açısından, bir css dokümanı açıp "stil.css" adıyla kaydediyorum.

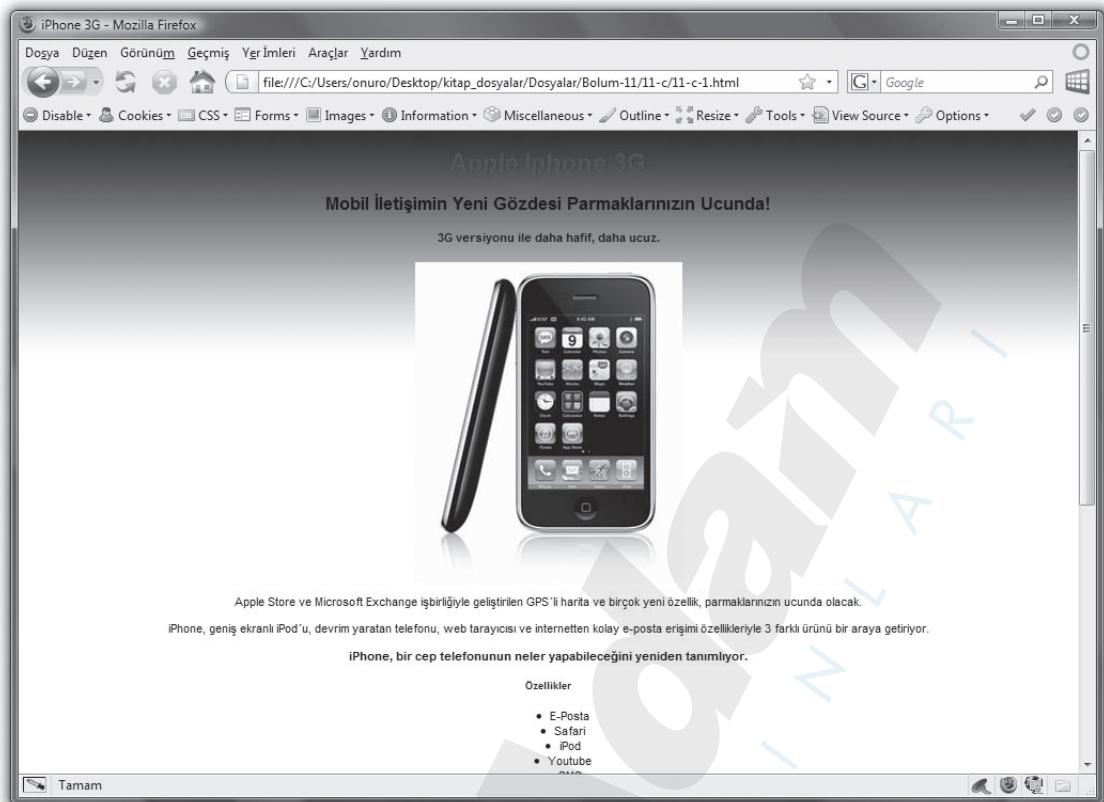
Şimdi bu stil dokümanını HTML dokümanımıza dahil edelim:

```
<link rel="stylesheet" type="text/css" href="stil.css" />
```

Dolayısıyla yapacağımız stillendirmeler "stil.css" dokümanımızda yer alarak, içindeki tanımlamalar HTML dokümanımızda değişiklikleri sağlıyor olacak.

Stil dosyamız hazır olduğuna göre, ilk satırlarımızı body elementimiz için açarak genel özellikleri tanımladıktan sonra arkaplan grafiğimizi yatay olarak tekrar edecek şekilde kendisine atayalım:

```
body {
margin:0; /* dış boşluğu sıfırla */
padding:0; /* iç boşluğu sıfırla */
text-align:center; /*Internet Explorer'da da anakutu öğesinin
ortalanmasını sağlamak için hazırlan*/
background:url(images/arkaplan.jpg) repeat-x; /* arkaplan
grafiğini yükle ve yatay tekrar et */
}
```



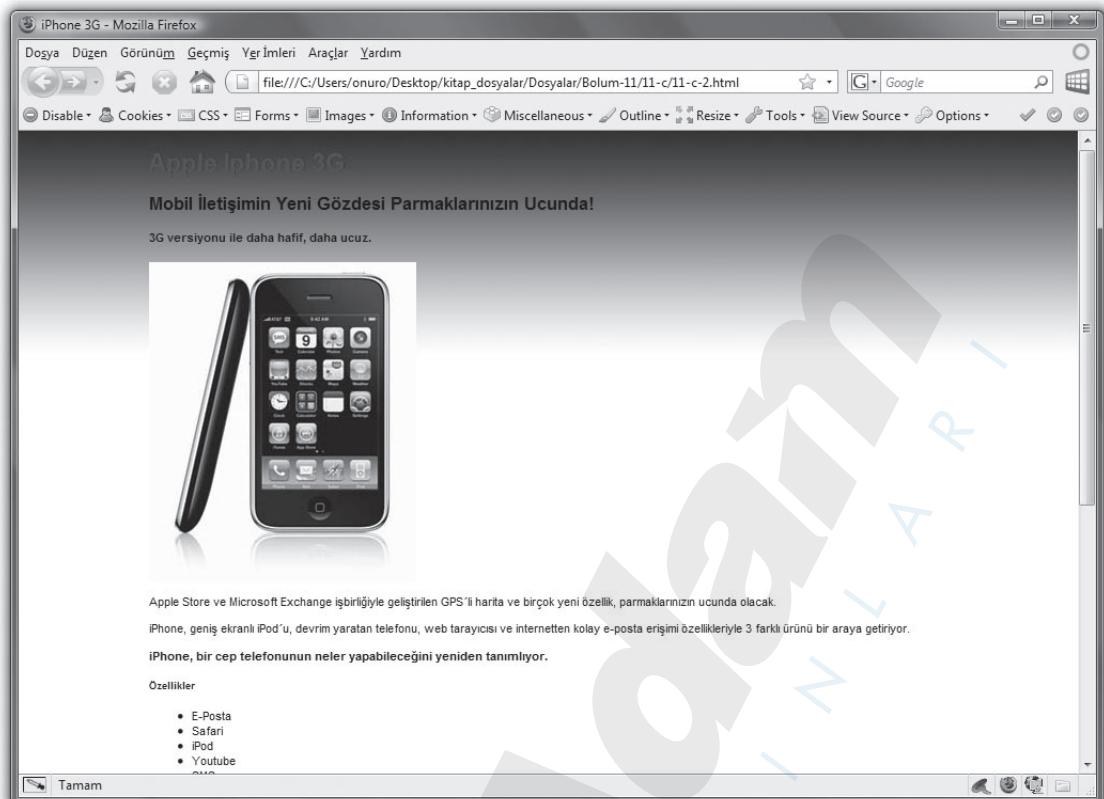
Figür 11-c-1

Ortalanınan anakutu içeriğimizi, kendisini biçimlerken tekrar sola alacağız. Ondan önce anakutu-muzun hangi özelliklere sahip olacağını belirleyelim:

1. Tasarım çalışmasındaki genişlik alanı 748 px oranına sahip. Bu durumda anakutu genişliği-miz de bu değere sahip olacak.
2. Boşluk kalmadan ortalanması için margin özelliğini kullanmamız gerekecek.

Şimdi anakutu özelliklerimize yukarıda tanımladığımız şekilde değerlerini girelim:

```
#anakutu {
    margin:0 auto; /* dış boşluğu sıfırla ve kutuyu dokümana ortala */
    text-align:left; /* ortalanmış içeriği tekrar sola al */
    padding:0; /* iç boşluğu sıfırla */
    width:748px; /* tasarımdaki alan genişliği */
}
```



Figür 11-c-2

Anakutumuzun genel özellikleriyle işimiz bitti.

Şu ana kadarki aşamayı dilseniz bir de Internet Explorer ile kontrol edelim:



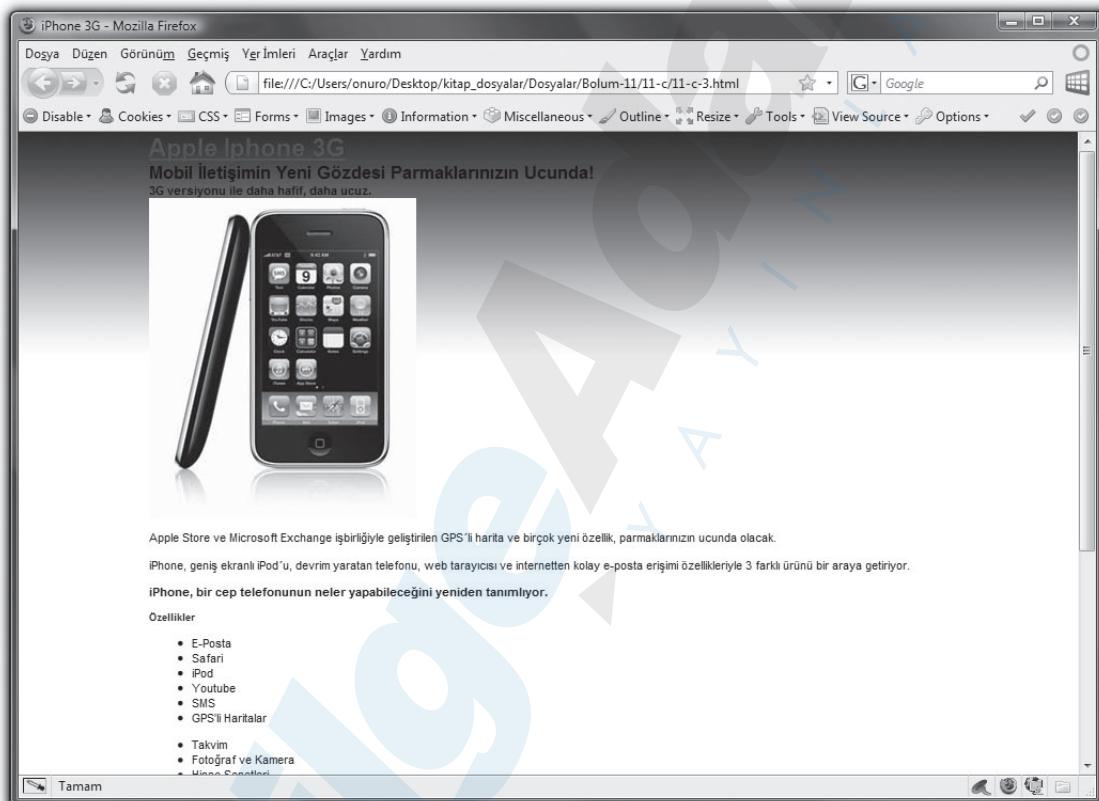
Görüntü 11-c-2

Her zamanki gibi, Internet Explorer'da bazı elementlerimizin boşluk ve metin değerleri farklı yorumlanıyor.

Şu an için metin boyutunun Internet Explorer tarafından farklı yorumlanması bize bir zararı yok. Ancak başlıklardaki boşlukları farklı yorumlaması (tarayıcı standartı boşluklarından ötürü) tasarım yapımızı bozabilir.

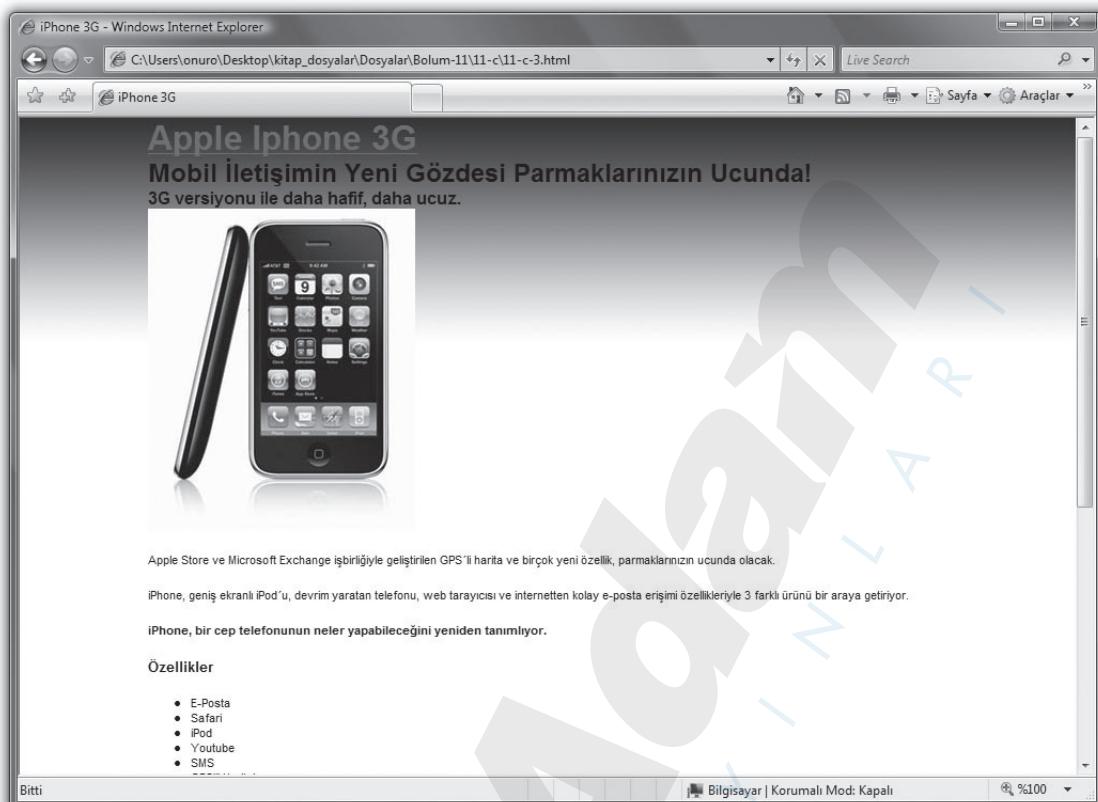
Bu yüzden işimizi saglama alalım ve tüm başlık öğelerinin boşluklarını sıfırlayalım:

```
h1, h2, h3, h4, h5, h6 {
margin:0;
padding:0;
}
```



Figür 11-c-3

Sonuca bir de Internet Explorer ile bakalım:



Görüntü 11-c-3

Yazıntılarının punto boyutu ve paragraf boşluk oranlarının farklılığı dışında tarayıcı ortak görünümünü sağladık.

Artık üst kutumuzu biçimlendirme aşamasına geçebiliriz.

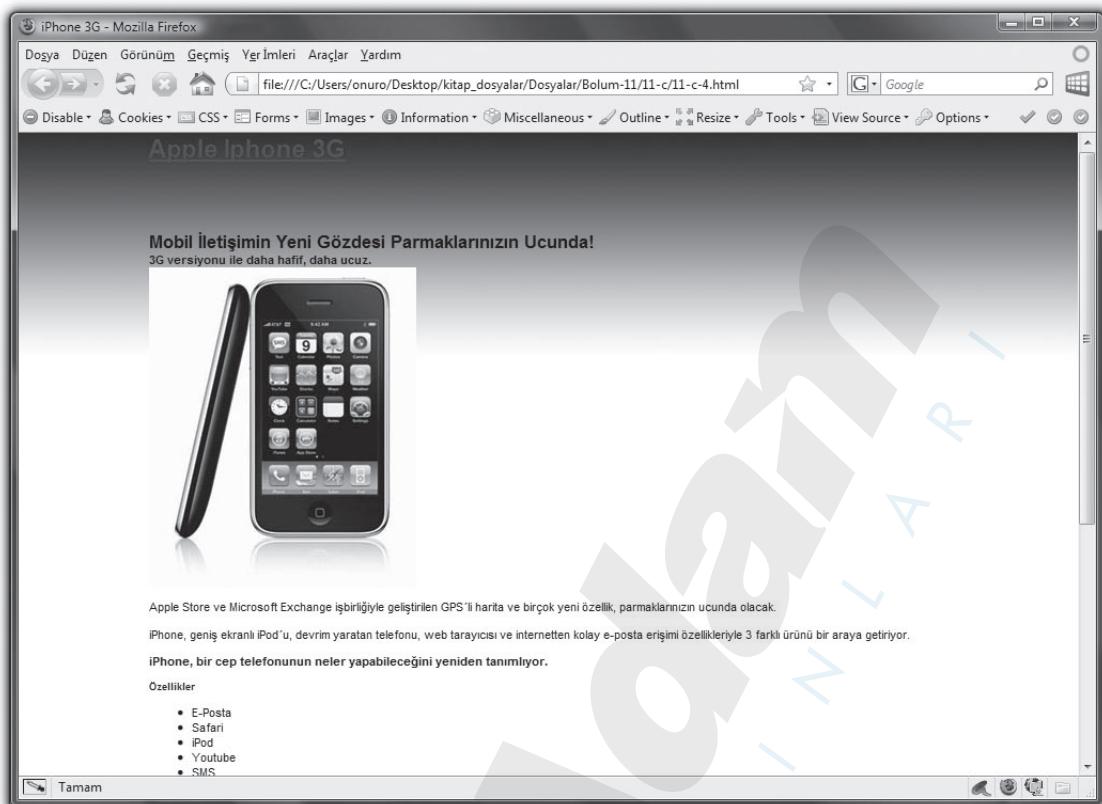
Üst kutumuz, içerik olarak bir başlık ve onun içinde bulunan bir linkten ibaret.

Photoshop'ta rehber çizgileri kullanarak üst kutu kapsama alanının yüksekliğini kontrol edecek olursak, aşağı yukarı **92 piksel** oranını üst kutumuza yükseklik olarak atamamız yeterli olacaktır.

Genişlik değeri tarayıcı standardında %100 olduğu, ve sınırları en fazla **anakutu** genişliğine kadar dayandığı için kendisine ayrıca bir genişlik vermemize gerek yok.

Üst kutumuz için yükseklik değerini atayalım:

```
#ust_kutu {
    height:92px;
}
```

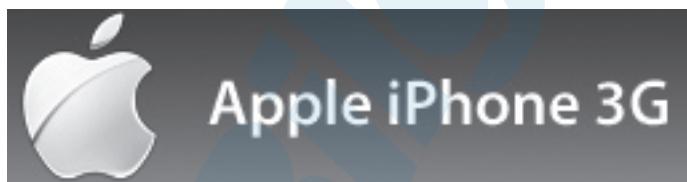


Figür 11-c-4

Üst kutu yüksekliğimiz, böylece kendi kapsam alanı altında kalacak öğeleri de aşağı kaydırılmış oldu. Bu boşluğa başlığımızın içindeki link elementimizi stillendirirken ihtiyacımız olacak.

Şimdi yapmamız gereken, h1 başlığımızdan çok onun içindeki a elementini biçimlemek. Zira tıklanabilir alan ve arkaplan grafiği olacak apple logosu ve başlığı link elementimiz ile doğrudan bağlantılı.

Başlık link elementimiz yerini alacak grafiğimizi Photoshop içinde kestikten sonra "iphone_baslik" adıyla kaydediyoruz. Dosyamız 267 piksel genişliğinde ve 66 piksel yüksekliğinde.



Görüntü 11-c-4

Grafiğimizi link elementimize arkaplan olarak atamadan önce, karakteristik özelliklerini ve tıklanacak alan olarak kutu yapısı kazanması için `display:block` değerini atayalım:

```
#ust_kutu h1 a {
    display:block;
    width:257px;
    height:66px;
}
```



Figür 11-c-5

Böylece link elementimizin fiziki yapısı grafik arkaplana hazır hale geldi.

Arkaplan grafiğimizi uygulayalım:

```
#ust_kutu h1 a {
    display:block;
    width:257px;
    height:66px;
    background:url(images/iphone_baslik.jpg) no-repeat;
}
```

Sıra link metnimizi ortadan kaldırmaya geldi.



Figür 11-c-6

Bu işlemi, paragraf metin boşluğu atamada kullandığımız `text-indent` özelliğine negatif değer uygulayarak gerçekleştireceğiz:

```
#ust_kutu h1 a {
    display:block;
    width:257px;
    height:66px;
    background:url(images/iphone_baslik.jpg) no-repeat;
    text-indent:-9999px; }
```



Figür 11-c-7

Üst bölümle işimiz neredeyse bitti.

Şimdi, link elementimizi tepeden biraz aşağıya almamız gerekecek. Photoshop'taki tasarımda rehber çizgisini tepe üst hızası ile logomuz arasına çektiğimizde 25 piksellik boşluk olduğu ortaya çıkıyor.

Bu değeri link elementimizin üst boşluk değeri olarak atayabiliriz:

```
#ust_kutu h1 a {
    display:block;
    width:257px;
    height:66px;
    background:url(images/iphone_baslik.jpg) no-repeat;
    text-indent:-9999px;
    margin-top:25px;
}
```



Figür 11-c-8

İşimiz bitti görünüyor. Ancak siz de fark ettiniz mi, üst kutu ve onu kapsayan anakutu dahil tüm öğeler de aslında link elementimize atadığımız 25 piksel üst boşluk değerinden sonra aşağı kaymış durumda. Bu az sonra biçimleyeceğimiz orta kutunun dikey hızası için önemli bir sorun oluşturabilir:



Görüntü 11-c-5

Dolayısıyla link veya başlık (h1) elementimize margin-top biçimlemesi yapmak problem haline geldiğinden, bir başka yöntemle bu boşluğu elde etmemiz gerekiyor.

Biraz mantığımıza başvurursak, a veya h1 elementimize margin-top vermek yerine üst kutu elementimize iç boşluk (padding-top) vermek işimizi görür mü? Elbette görür.

padding değerleri margin'e göre biraz daha farklı yorumlandılarından, link elementimizden margin değerinin satırını sildikten sonra üst kutu padding-top değerine 25 piksel girerken, yüksekliğini de 64 piksel değerine ayarlıyoruz:

```
#ust_kutu {
height:64px;
padding:25px;
}
```

**Figür 11-c-9**

Sorunumuzu çözerek üst kutuya ilgili tüm stillendirmeyi tamamlamış olduk.

Bir sonraki adımda, orta kutumuzu biçimlendiriyor olacağız.

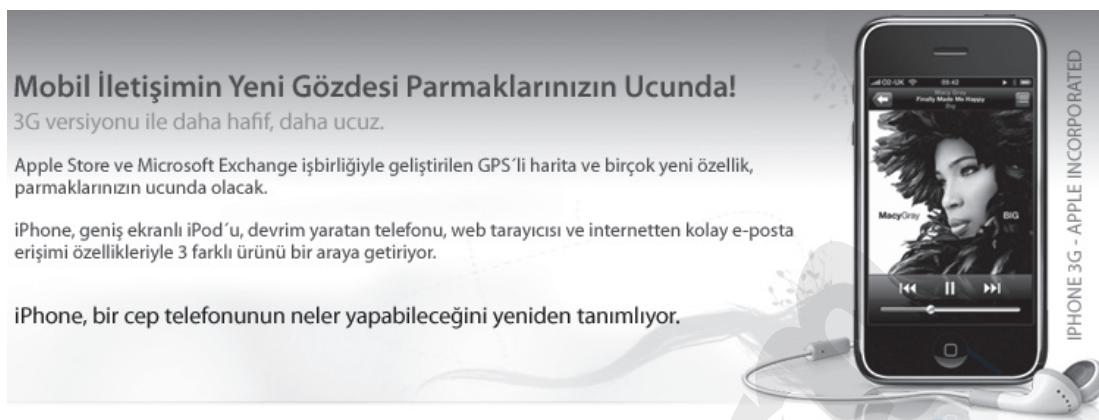
11.d Orta Kısmı Stillendirmek

Bildiğiniz gibi, üst kutudan hemen sonra gelen orta kutumuz iki başlık ve birkaç paragraftan oluşuyor.

Normal şartlarda bu kutu içindeki her element için “görsel yerdeğişim” uygulamaya kalkarsak işimiz hem uzun sürecek, hem de oldukça zahmetli bir çalışma içine gireceğiz.

Bunun yerine, orta kutu elementimiz içindeki öğeleri kapatıp (display : none), yalnızca grafik arkaplandan oluşan bir kutu haline getirerek hızlı bir şekilde stillendirme işlemini tamamlayabiliriz.

Tasarım çalışmasındaki orta bölüm içeriğini kestikten sonra “orta_arkaplan.jpg” adıyla kaydediyoruz. Bu grafik 748 piksel genişliğe ve 289 piksel yüksekliğe sahip:

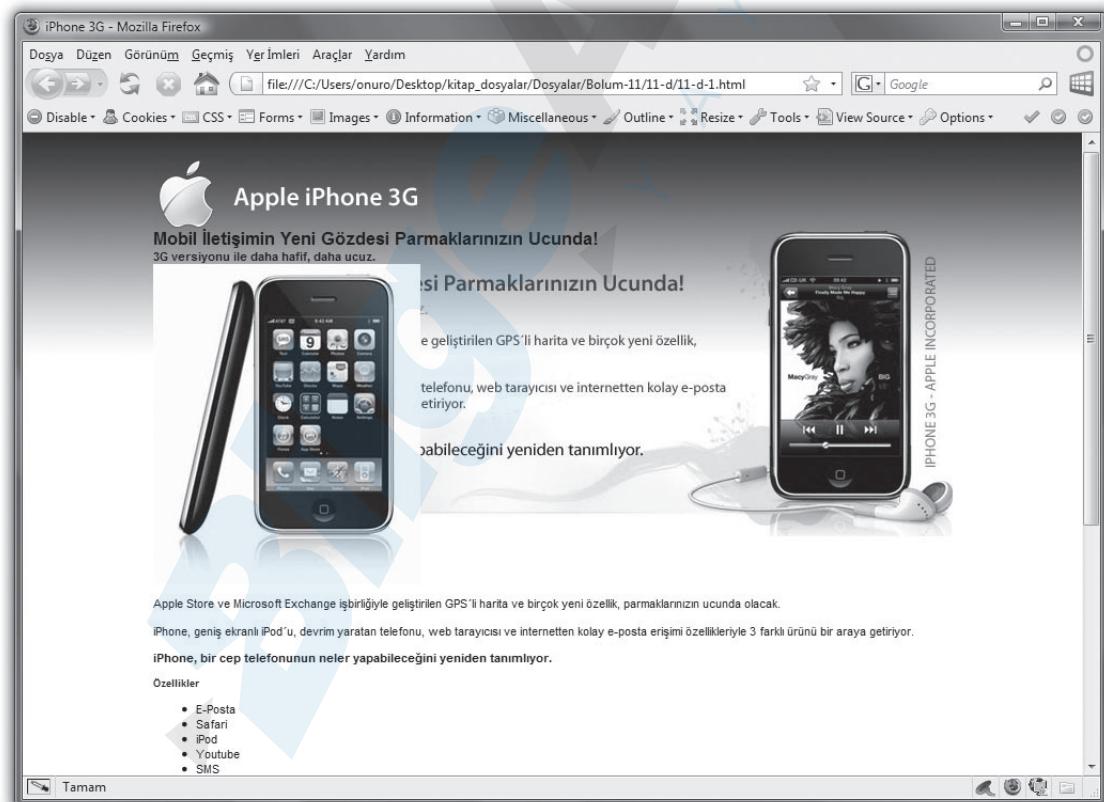
**Görüntü 11-d-1**

Bu grafik bizim orta kutumuzun ta kendisi olacak.

Dolayısıyla arkaplanı orta kutu div elementimizin selektörüne atadıktan sonra, orta kutu içeriğindeki elementleri `display:none` ile gizleyeceğiz.

Öncelikle CSS dokümanımızda orta kutu selektörümüzü oluşturup arkaplan grafiğimizi atayalım:

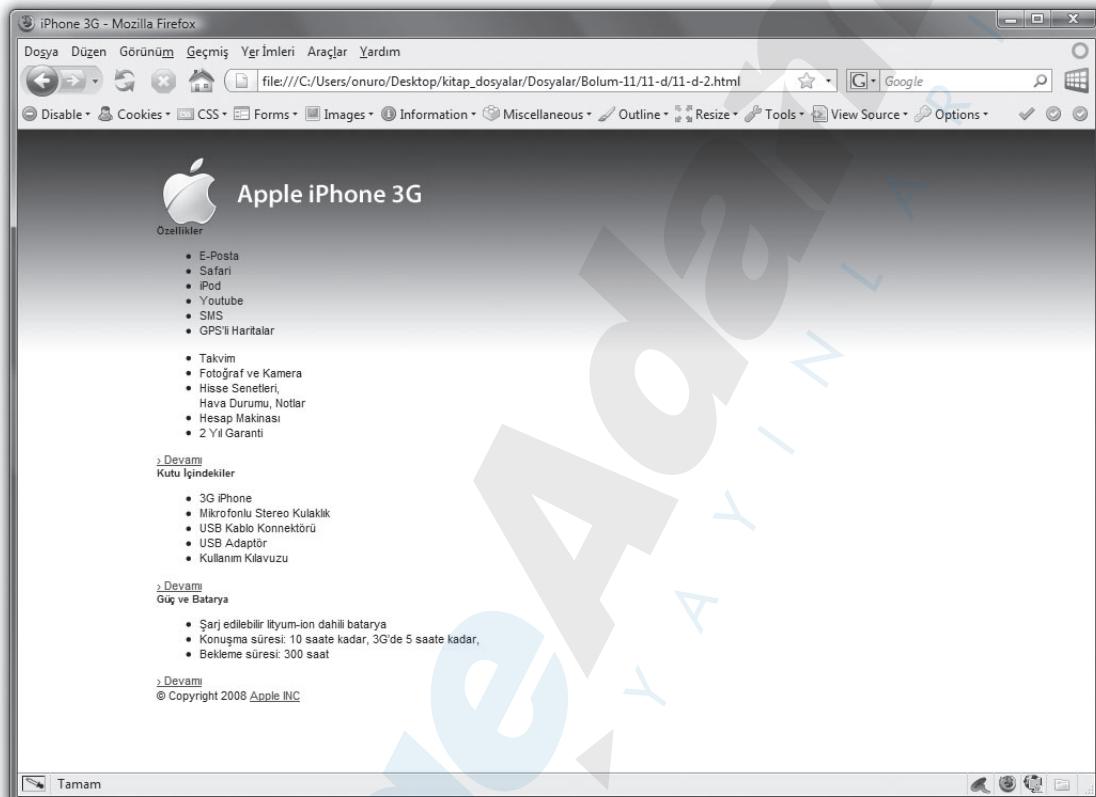
```
#orta_kutu {
background:url(images/orta_arkaplan.jpg) no-repeat;
}
```

**Figür 11-d-1**

Görüldüğü gibi orta kutu arkaplanımız, kutu içeriğimizin arkasına yerleştirdi ve tekrar etmeden pozisyonunu koruyor.

Şimdi sıra orta kutu içindeki tüm elementleri ortadan kaldırırmaya geldi:

```
#orta_kutu h2, #orta_kutu h4, #orta_kutu img, #orta_kutu p {
    display:none;
}
```



Figür 11-d-2

Orta kutu içindeki tüm öğelerin yok edilmesinden ötürü, orta kutumuzun tarayıcı standarı yükseklüğü de kesilmiş görünüyor. Bu nedenle de halen orada olan orta kutumuzu göremiyoruz.

Orta kutumuza, grafik arkaplanımızın yüksekliğini atadığımızda bu sorun çözülmüş olacak:

```
#orta_kutu h2, #orta_kutu h4, #orta_kutu img, #orta_kutu p {
    display:none;
}
```

Şimdiden oldukça etkileyici görünüyor. Stilsiz olarak içeriğini görüntülediğimiz orta kutumuz, stilli haliyle son derece sık bir sunuma sahip.



Figür 11-d-3

İçerik kutumuz ve içindeki liste öğelerine tasarımdaki görünümü uygulamakla devam edeceğiz.

11.e İçerik Kısımını Stillendirmek

Photoshop'taki rehber çizgilerle yapılan ölçüme göre 170 piksel yüksekliğe sahip olan içerik kutumuz, 3 alt kutudan ve içeriklerindeki liste öğelerinden oluşuyor:



Görüntü 11-e-1

Sıkıntısız ve hızlı bir şekilde gidebilmek için, şöyle bir plan uygulayabiliriz:

1. İçerik kutusunun arkaplan grafik ve yükseklik özelliklerini selektörünü açarak uygularız.
2. Kutudaki 3 alt kutu içinde bulunan listelerin (ul) ortak görünüm standartlarını stillendirerek ayarlarız.
3. 3 kutunun arkaplan grafik ve genişlik atamalarını yaparız.
4. İçerik kutusundaki link öğelerini stillendiririz.

İşte Photoshop yardımıyla içerik kutumuzun arkaplanını oluşturacak kısmı kestikten sonra “icerik-arkaplan.jpg” adıyla kaydederek “images” klasörümüze kaydederek başlayabiliriz:

Görüntü 11-e-2

İçerik kutu yükseklik değerini girdikten sonra öğemize arkaplan grafiğini atayalım:

```
#icerik_kutusu {  
height:170px;  
background:url(images/icerik-arkaplan.jpg) no-repeat bottom;  
}
```



Figür 11-e-1

İçerik kutumuza verdigimiz yükseklik, alt kutumuzun da hemen onun altına yerleşmesini sağladı.

Listelerin alt alta gelmesinden kaynaklanan karışıklığı sıkıntılı etmiyoruz, az sonra onların da yan yana gelmelerini sağlayacağız.

Şimdi dilerken tüm tarayıcılarda listelerimizin üstündeki başlıkların ortak görünümü için metin boyutlarını ayarlayalım ve biraz alt boşluk verelim:

```
#icerik_kutusu h5 {  
font-size:1.1em;  
margin-bottom:10px;  
}
```



Figür 11-e-2

Aslında alt boşluk oranı bir önceki durumla pek farklılık göstermedi. Ancak bu orani vermemiş olsaydık, normal şartlarda kalıtsallıkla sıfır boşluğa sahip başlıklarımız, bir adım sonra sıfırlaya-cağımız liste (ul) boşlukları nedeniyle listelerle birbirine yapışacaktı.

Şimdi de listelerimizin iç ve dış boşluklarını sıfırlayarak başlık hizalarına sağlayalım:

```
#icerik_kutusu ul {
margin:0;
padding:0; }
```

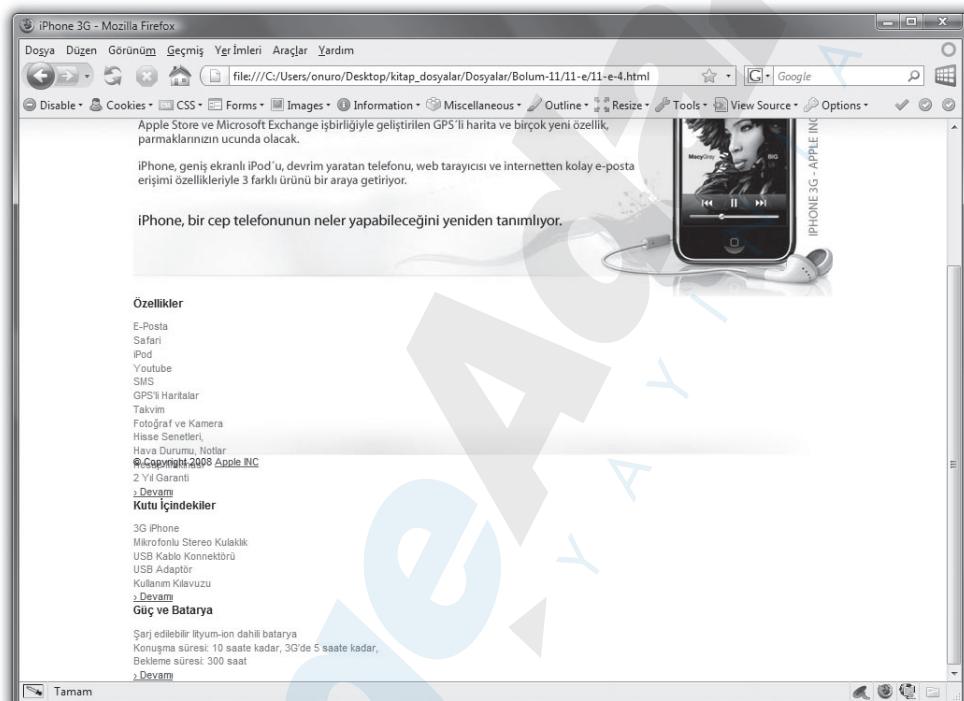


Figür 11-e-3

Listelerimiz başlıklarımızla aynı hizaya geldi ve muhtemel sorun üretici tarayıcı standarı boşluklarından kurtuldular.

Sıra içerik kutumuzdaki liste öğelerinin (`ul>li`) görünümünü biçimlemeye geldi. Tasarımımızda gri renkte görüntülenen liste öğelerimizi bu şekilde stillendirerek satır aralarını tasarıma uygun bir bağıl orana (`em`) getirerek imleçlerini kaldırıyoruz:

```
#icerik_kutusu ul li {
list-style-type:none;
color:#777;
line-height:1.3em;
}
```



Figür 11-e-4

İç kutu genel stillendirmelerle işimiz neredeyse bittiğine göre, artık iç kutumuzun kapsadığı 3 id selektörlü kutumuzu stillendirmeye başlayabiliriz.

Tasarım çalışmamızda, bu kutuların her biri kendilerine ait genişliklere sahip. Ancak hepsinin bazı ortak özellikleri var:

1. Her biri sola yaslanıyor (bu `float` uygulayacağımız demektir).
2. Arkaplan grafikleri olacak ikonların genişliğini de dahil edersek, iç kutular içindeki tüm başlıklar ve listeler belli bir boşluk oranından sonra başlıyor (**72 piksel**).
3. Her kutunun genişliği bittikten sonra belli bir oranda boşluk var (`margin-left` veya `margin-right`).

O halde bu özellikleri kutularımızın ortak özellikleri olarak uygulayalım:

```
#ozellikler, #kutu_icindekiler, #guc_ve_batarya {
    float:left;
    padding-left:72px;
    margin-right:10px;
}
```



Figür 11-e-5

İç kutu bölümümüz biraz şekillenmeye başladı.

Şimdi liste öğelerimize belirli stiller atayarak 3 kutumuzda da ortak görünüm elde etmeye çalışalım.

Tasarımı baz aldığımdızda liste öğelerinin sola yaslandığını ve belli bir genişlik oranına sahip olduğunu ve altlarındaki linklerin kendilerine yapışmadığını, belirli bir boşluktan başladığını gördük.

Görünüm farklarını dokümanımızda stillendirme yaparak tasarımdaki görünümle eş hale getirmeye çalışalım:

```
#ozellikler ul, #kutu_icindekiler ul, #guc_ve_batarya ul {
    min-width:100px;
    max-width:110px;
    margin-bottom:10px;
    float:left;
}
```



Figür 11-e-6

Her ne kadar linklerimizin bulunduğu div öğeler liste öğelerin float özelliği nedeniyle yanlarına yapışarak görünümü bozmamışlarsa da, bu durum için risk almayıp link div öğelerin sol taraflarını temizlememiz gerekiyor.

Bu işlem için, daha evvel de kullanmış olduğumuz clear özelliğinden yararlanacağız:

```
#ozellikler ul, #kutu_icindekiler ul, #guc_ve_batarya ul {
    min-width:100px;
    max-width:110px;
    margin-bottom:10px;
    float:left;
}
```



Figür 11-e-7

İç kutuların ortak görünüm tanımlamalarıyla işimiz bitti.

Bundan sonraki adımlarda kutuların kendi selektörlerini açıp görünümlerini biçimlemeye başlayabiliriz.

Öncelikle, kutuların kendi genişlik ve arkaplan özelliklerini tanımlamamız gerekiyor. Bunun için, her kutunun arkaplanını kendi genişlik oranına göre Photoshop'ta kesip uygun isimlendirmelerle "images" klasörümüze kaydediyoruz:



Görüntü 11-e-3: ozellikler_arkaplan.jpg (312 piksel genişliğinde)



Görüntü 11-e-4: kutu_icindekiler_arkaplan.jpg (212 piksel genişliğinde)



Görüntü 11-e-5: kutu_icindekiler_arkaplan.jpg (59 piksel genişliğinde, ancak kutumuz bundan daha geniş olacak)

Öncelikle özellikler kutumuzla başlayabiliriz. Genişlik özelliğine grafik arkaplanındaki gibi 212 piksel vermemiz gerektiğini düşünmek mümkün, ancak daha evvel 3 kutumuza uyguladığımız kalıtsal biçimlemelerde 72 piksellik sol iç boşluk (padding-left) belirlediğimizden, bunun div kutularımıza eklemeye yapacağımı da hatırlamamız gerekiyor.

Özellikler kutumuza arkaplan grafiğini atayarak, genişliğini verirken sahip olduğu değerden 72 piksel değerini eksilterek uyguluyoruz:

```
#ozellikler {
    width:242px;
    background:url(images/ozellikler_arkaplan.jpg) no-repeat;
}
```



Figür 11-e-8

Özellikler kutumuzla işimiz bitti.

`kutu_icindekiler` div elementimizle devam edecek olursak, onun da arkaplanını atarken sahip olduğu değerden 72 piksel değerini eksiltmemiz gerekiyor:

```
#kutu_icindekiler {
    width:142px;
    background:url(images/kutu_icindekiler_arkaplan.jpg) no-repeat;
}
```



Figür 11-e-9

Geriye son olarak güç ve batarya kutumuz kaldı.

Bu kutu için **115 piksellik** genişlik yeterli olacaktır. Zira fiziki genişliği 72 piksellik iç boşluk da eklenince **178 piksel** genişliğiyle görüntülenecektir.

Arkaplan ve genişlik özelliklerini bu kutumuzu da uyguluyoruz:

```
#guc_ve_batarya {
    width:115px;
    background:url(images/guc_ve_batarya_arkaplan.jpg) no-repeat; }
```



Figür 11-e-10

Son oldukça yaklaştık.

Şimdi biraz geriye gidip, içerik kutumuzda stilsiz kalan linklerimizi renklendirmemiz gerekiyor.

Tasarımımızda linklerimiz siyah renkte ve altlarında çizgi (underline) bulunmuyor. Biz stillendiğinde bunları uygularken ek olarak hover durumu için alt çizgileri geri getirelim:

```
#icerik_kutusu a {
    color:black;
    text-decoration:none;
}

#icerik_kutusu a:hover {
    text-decoration:underline;
}
```



Figür 11-e-11

Dokümanınızın alt_kutu bölümünü saymazsa, neredeyse birebir olarak Photoshop tasarımlımızla aynı görünümde sahip sık bir CSS destekli HTML dokümanı elde ettik.

Stillendirmemizin son aşamasında, alt_kutu özelliklerimizi tanımlıyor olacağız.

11.f Alt Kısmı Stillendirmek

Ana kutumuz kapanmadan önceki son elementimiz, dokümanımızın görsel olarak sona erdiğini haber veren alt_kutu elementimiz.

Yapmamız gereken, bu kutumuza biraz üst ve sol iç boşluk vererek, metin ve link rengini gri olarak belirleyip yazıtipi boyutunu biraz küçültmek:

```
#alt_kutu {
padding:10px 0 0 10px;
}

#alt_kutu, #alt_kutu a {
color:#999;
font-size:0.9em;
}
```



Figür 11-f-1

Harika görünüyor, değil mi?

Çalışmamızın Internet Explorer ile aynı görünümüne sahip olup olmadığını son kez kontrol edelim:

**Görsüntü 11-f-1**

Göze batmayacak kadar ufak piksel kaymaları dışında birebir görünüm elde ettik.

Böylece, tasarlanmış olan bir sayfamızı, CSS yardımıyla hem sık hem de erişilebilir bir formatta hayata geçirmiş olduk.

HTML ve CSS dokümanımızı W3C (<http://www.w3.org>) onayına sokacak olursak, hem CSS (<http://jigsaw.w3.org/css-validator/>), hem de HTML (<http://validator.w3.org/>) kodlarımızın da onaylı olduğunu, semantik olarak da erişilebilir bir sonuç elde ettiğimizi görürüz.



A

**Tam Liste
CSS Özellik
Tablosu**

Tam Liste CSS Özellik Tablosu

Worldwide Web Konsorsiyumu'nun (W3C) desteklediği CSS selektör özellikleri listesidir.

Kaynak: (<http://www.w3.org/TR/CSS21/propidx.html>)

Özellik	Alabilecegi Değerler	Standart Değer	Uygulanabilecek Elementler (Varsayılan: hepsi)	Kalıtsallık
'azimuth'	<angle> [[left-side far-left left center-left center center-right right far-right right-side] behind] leftwards rightwards inherit	center		var
'background-attachment'	scroll fixed inherit	scroll		yok
'background-color'	<color> transparent inherit	transparent		yok
'background-image'	<uri> none inherit	none		yok
'background-position'	[[<percentage> <length> left center right] [<percentage> <length> top center bottom]?] [[left center right] [top center bottom]] inherit	0% 0%		yok
'background-repeat'	repeat repeat-x repeat-y no-repeat inherit	repeat		yok
'background'	['background-color' 'background-image' 'background-repeat' 'background-attachment' 'background-position'] inherit	Her bir özelliğe bakın		yok
'border-collapse'	collapse separate inherit	separate	'table' ve 'inline-table' elementlerine	var
'border-color'	[<color> transparent]{1,4} inherit	Her bir özelliğe bakın		yok
'border-spacing'	<length> <length>? inherit	0	'table' ve 'inline-table' elementlerine	var
'border-style'	<border-style>{1,4} inherit			yok

Özellik	Alabilecegi Değerler	Standart Değer	Uygulanabilecek Elementler (Varsayılan: hepsi)	Kalıtsallık
'border-top' 'border-right' 'border-bottom' 'border-left'	[<border-width> <border-style> 'border-top-color'] inherit			yok
'border-top-color' 'border-right-color' 'border-bottom-color' 'border-left-color'	<color> transparent inherit			yok
'border-top-style' 'border-right-style' 'border-bottom-style' 'border-left-style'	<border-style> inherit	none		yok
'border-top-width' 'border-right-width' 'border-bottom-width' 'border-left-width'	<border-width> inherit	medium		yok
'border-width'	<border-width>{1,4} inherit			yok
'border'	[<border-width> <border-style> 'border-top-color'] inherit			yok
'bottom'	<length> <percentage> auto inherit	auto	Pozisyonlanmış elementlere	yok
'caption-side'	top bottom inherit	top	'table-caption' elementlerine	var
'clear'	none left right both inherit	none	block-level elementlere	yok
'clip'	<shape> auto inherit	auto	Absolute konumlanmış elementlere	yok
'color'	<color> inherit			var
'content'	normal none [<string> <uri> <counter> attr(<identifier>) open-quote close- quote no-open-quote no-close-quote]+ inherit	normal	:before ve :after pseudo- elementlerine	yok
'counter- increment'	[<identifier> <integer>?]+ none inherit	none		yok

Özellik	Alabilecegi Değerler	Standart Değer	Uygulanabilecek Elementler (Varsayılan: hepsi)	Kalıtsallık
'counter-reset'	[<identifier> <integer>?]+ none inherit	none		yok
'cue-after'	<uri> none inherit	none		yok
'cue-before'	<uri> none inherit	none		yok
'cue'	['cue-before' 'cue-after'] inherit			yok
'cursor'	[[<uri> ,]* [auto crosshair default pointer move e-resize ne-resize nw-resize n-resize se-resize sw-resize s-resize w-resize text wait help progress]] inherit	auto		var
'direction'	ltr rtl inherit	ltr	Tüm elementler	var
'display'	inline block list-item run-in inline-block table inline-table table-row-group table-header-group table-footer-group table-row table-column-group table-column table-cell table-caption none inherit	inline		yok
'elevation'	<angle> below level above higher lower inherit	level		var
'empty-cells'	show hide inherit	show	'table-cell' elementlerine	var
'float'	left right none inherit	none	hepsine	yok
'font-family'	[[<family-name> <generic-family>] [, <family-name> <generic-family>]*] inherit			var
'font-size'	<absolute-size> <relative-size> <length> <percentage> inherit	medium		var
'font-style'	normal italic oblique inherit	normal		var

Özellik	Alabilecegi Değerler	Standart Değer	Uygulanabilecek Elementler (Varsayılan: hepsi)	Kalıtsallık
'font-variant'	normal small-caps inherit	normal		var
'font-weight'	normal bold bolder lighter 100 200 300 400 500 600 700 800 900 inherit	normal		var
'font'	[['font-style' 'font-variant' 'font-weight']? 'font-size' [/ 'line-height']? 'font-family'] caption icon menu message-box small-caption status-bar inherit	Her bir özelliğe bakın		var
'height'	<length> <percentage> auto inherit	auto		yok
'left'	<length> <percentage> auto inherit	auto	position özelliği kullanılmış elementlere	yok
'letter-spacing'	normal <length> inherit	normal		var
'line-height'	normal <number> <length> <percentage> inherit	normal		var
'list-style-image'	<uri> none inherit	none	'display: list-item' özelliği atanmış elementlere	var
'list-style-position'	inside outside inherit	outside	'display: list-item' özelliği atanmış elementlere	var
'list-style-type'	disc circle square decimal decimal-leading-zero lower-roman upper-roman lower-greek lower-latin upper-latin armenian georgian lower-alpha upper-alpha none inherit	disc	'display: list-item' özelliği atanmış elementlere	var
'list-style'	['list-style-type' 'list-style-position' 'list-style-image'] inherit	Her bir özelliğe bakın	'display: list-item' özelliği atanmış elementlere	var
'margin-right' 'margin-left'	<margin-width> inherit	0		yok
'margin-top' 'margin-bottom'	<margin-width> inherit	0		yok

Özellik	Alabilecegi Değerler	Standart Değer	Uygulanabilecek Elementler (Varsayılan: hepsi)	Kalıtsallık
'margin'	<margin-width>{1,4} inherit	Her bir özelliğe bakın		yok
'max-height'	<length> <percentage> none inherit	none		yok
'max-width'	<length> <percentage> none inherit	none		yok
'min-height'	<length> <percentage> inherit	0		yok
'min-width'	<length> <percentage> inherit	0		yok
'orphans'	<integer> inherit	2		var
'outline-color'	<color> invert inherit	invert		yok
'outline-style'	<border-style> inherit	none		yok
'outline-width'	<border-width> inherit	medium		yok
'outline'	['outline-color' 'outline-style' 'outline-width'] inherit	Her bir özelliğe bakın		yok
'overflow'	visible hidden scroll auto inherit	visible		yok
'padding-top' 'padding-right' 'padding-bottom' 'padding-left'	<padding-width> inherit	0		yok
'padding'	<padding-width>{1,4} inherit	Her bir özelliğe bakın		yok
'page-break-after'	auto always avoid left right inherit	auto	block-level elementlere	yok
'page-break-before'	auto always avoid left right inherit	auto	block-level elementlere	yok
'page-break-inside'	avoid auto inherit	auto	block-level elementlere	var

Özellik	Alabilecegi Değerler	Standart Değer	Uygulanabilecek Elementler (Varsayılan: hepsi)	Kalıtsallık
'pause-after'	<time> <percentage> inherit	0		yok
'pause-before'	<time> <percentage> inherit	0		yok
'pause'	[[<time> <percentage>]{1,2}] inherit	Her bir özelliğe bakın		yok
'pitch-range'	<number> inherit	50		var
'pitch'	<frequency> x-low low medium high x-high inherit	medium		var
'play-during'	<uri> [mix repeat]? auto none inherit	auto		yok
'position'	static relative absolute fixed inherit	static		yok
'quotes'	[<string> <string>]+ none inherit	Kullanıcı temsilcisine bağlıdır		var
'richness'	<number> inherit	50		var
'right'	<length> <percentage> auto inherit	auto	position özelliği kullanılmış elementlere	yok
'speak-header'	once always inherit	once	Table header bilgisi girilmiş olan elementlere	var
'speak-numeral'	digits continuous inherit	continuous		var
'speak-punctuation'	code none inherit	none		var
'speak'	normal none spell-out inherit	normal		var
'speech-rate'	<number> x-slow slow medium fast x-fast faster slower inherit	medium		var
'stress'	<number> inherit	50		var
'table-layout'	auto fixed inherit	auto		yok

Özellik	Alabilecegi Değerler	Standart Değer	Uygulanabilecek Elementler (Varsayılan: hepsi)	Kalıtsallık
'text-align'	left right center justify inherit	Eğer 'direction' değeri 'ltr' ise 'left' ve 'direction' değeri 'rtl' ise 'right' olarak davranan isimsiz bir değer.		var
'text-decoration'	none [underline overline line-through blink] inherit	none		yok (bkz. prose)
'text-indent'	<length> <percentage> inherit	0		var
'text-transform'	capitalize uppercase lowercase none inherit	none		var
'top'	<length> <percentage> auto inherit	auto	position özelliği kullanılmış elementlere	yok
'unicode-bidi'	normal embed bidi-override inherit	normal		yok
'vertical-align'	baseline sub super top text-top middle bottom text-bottom <percentage> <length> inherit	baseline		yok
'visibility'	visible hidden collapse inherit	visible		var
'voice-family'	[<specific-voice> <generic-voice> ,]* [<specific-voice> <generic-voice>] inherit	Kullanıcı temsilcisine bağlıdır		var
'volume'	<number> <percentage> silent x-soft soft medium loud x-loud inherit	medium		var
'white-space'	normal pre nowrap pre-wrap pre-line inherit	normal		var
'widows'	<integer> inherit	2		var

Özellik	Alabilecegi Değerler	Standart Değer	Uygulanabilecek Elementler (Varsayılan: hepsi)	Kalıtsallık
'width'	<length> <percentage> auto inherit	auto		yok
'word-spacing'	normal <length> inherit	normal		var
'z-index'	auto <integer> inherit	auto		yok