# ***CSS DERSLERİ***

## CSS Nedir?

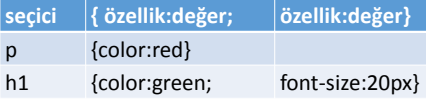
CSS, Cascade Style Sheet kelimelerinin kısaltılmışıdır. HTML komutlarını biçimlendirmek için kullanılırlar. Örneğin, CSS kullanarak <p> tagının içindeki metinlerin yazı tipini, rengini, boyutunu ve diğer birçok görsel özelliğini değiştirebilirsiniz. Style kelimesi Türkçe’ de ki biçem kelimesine karşılık gelir. Biçem kabaca, ayarlar topluluğu olarak tanımlanır. Yani, bazı ayarlar yapıyoruz, bu ayarlara bir isim veriyoruz ve bu biçemi değişik nesnelere uygulayarak nesnenin görüntüsünü değiştiriyoruz.

## CSS Yazım Kuralları

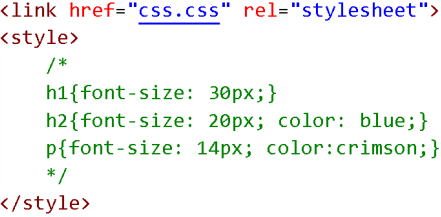
CSS komutları 3 parçadan oluşur. Bunlar:

* HTML Komutu (Selector)
* Özellik (Property)
* Değer (Value)

Tipik bir CSS komutunun yapısı aşağıdaki gibidir:



* Eğer CSS kodları içine açıklama metinleri eklemek istiyorsanız aşağıdaki sözdizimini kullanabilirsiniz:



* + /\* Açıklama metni \*/

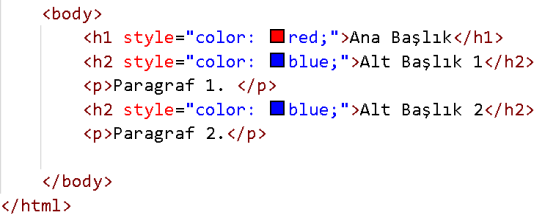
## CSS Kodları Nereye Yazılır?

CSS kodlarını 3 farklı yere yazabilirsiniz:

1. HTML tagının içine (satır içi / inline)
2. Sayfanın head bölümüne (dahili / internal)
3. Bir dosyaya (harici / external)

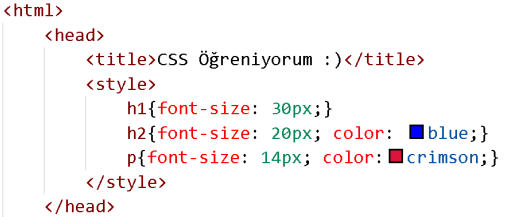
## Satır İçi (inline) CSS Kodları

* CSS kodları style özelliği kullanılarak doğrudan komut içine yazılır.



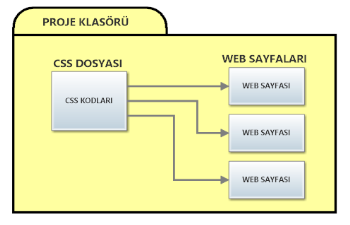
## Dahili (internal) CSS Kodları

* CSS kodları head bölümünde style etiketinin içine yazılır.



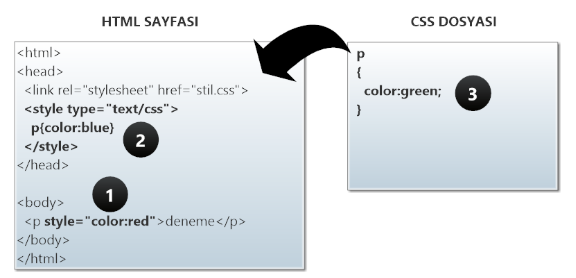
## Harici (external) CSS Kodları

* CSS kodları metin tabanlı bir dosya içine yazılır ve ihtiyaç duyulan her sayfaya bağlanır (link).
  + <link rel="stylesheet" href="stil.css" />



## Hangi CSS Kodu Öncelikli?

* HTML komutuna daha yakın olan CSS kodu önceliklidir.



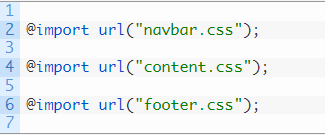
## @import Özelliği ve Kullanımı

Bu yazımızda Css’te import kullanımını öğreneceğiz. Örneğin bir tasarım yaptığımızı düşünelim. Tasarım için yazdığımız tüm Css kodlarını tek dosya halinde tutarsak ilerleyen zamanlarda boyutu çok büyüyeceği için biraz karışabilir. Neyi nerede yazdığımızı bulmakta zorlanabiliriz. Eğer tasarımda belirli kısımlar için yazdığımız Css kodlarını ayrı ayrı Css dosyalarına yazarsak daha düzenli olacağından yönetilebilirliği arttırmış oluruz.

Örneğin menü kısmı için yazdığımız Css kodlarını **navbar.css**, içerik kısmı için yazdığımız Css kodlarını **content.css**, footer kısmı içinde **footer.css** dosyalarına yazdığımızı düşünelim. Bu şekilde menü ile ilgili bir değişiklik yapma ihtiyacı olduğunda hemen navbar.css dosyasını açıp gerekli değişiklikleri yapabiliriz. Diğer türlü hepsi tek Css dosyasında olduğunda biraz daha karışık hale gelebiliyor.

Artık import kullanmanın avantajını öğrendiğimize göre nasıl kullanabileceğimize bakalım. Tüm ayrı ayrı oluşturduğumuz Css dosyalarını birleştirmek için **site.css** dosyası oluşturalım ve bu dosyanın içerisine bunları ekleyelim.

**site.css dosyasının içeriği**

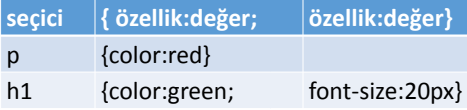
****

Önemli burada dosyalarınızın yolunu doğru yazdığınızdan emin olun. Daha sonra oluşturduğumuz bu dosyayı Html sayfamıza link vermeyi unutmayalım.



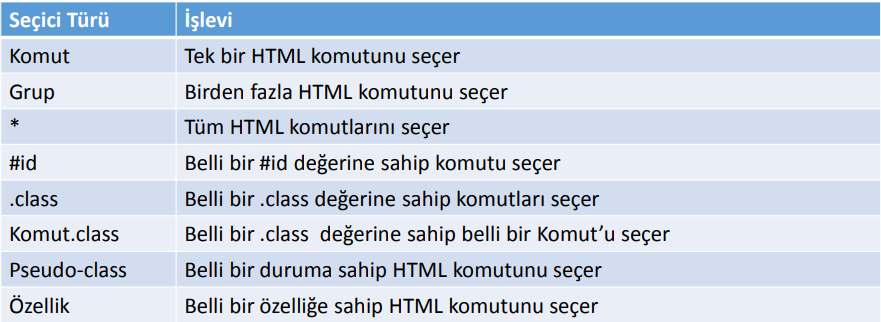
## Seçici İşlemeleri

* Biçimlendirilecek HTML komutlarını seçmeye yarar.
* CSS kodlarının en başına koyulur.
* Hangi HTML elemanları seçilirse CSS kodları onlara uygulanır. Tipik bir CSS komutunun yapısı aşağıdaki gibidir:



## Seçici Türleri

CSS komutundaki seçici biçimlendirme yapılacak HTML komutunun seçilmesini sağlar. Seçici türleri aşağıdaki gibidir:



## Etiket Seçici

* Tek bir HTML komutunun seçilmesini sağlar.
* Bir ya da daha fazla özellik kullanabilirsiniz.
* Özellikleri alt alta veya yan yana yazabilirsiniz.

## Tekil Seçici (#id)

* Belli bir #id değerine sahip HTML komutunu seçer.
* Tek bir tane etikete verilir.

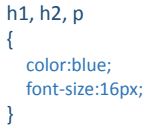
## Sınıf Seçici (.class)

* Belli bir .class değerine sahip HTML komutlarını seçer.
* CSS kodlarını birden fazla HTML komutuna uygulamak isterseniz sınıf seçiciyi kullanmalısınız.

## Evrensel Seçici

* Tüm HTML komutlarının seçilmesini sağlar.
* Yazılan kodlar sayfadaki tüm komutlara uygulanır.

## Grup Seçici

* Birden fazla HTML komutunun seçilmesini sağlar.
* Yazılan CSS kodları tüm HTML komutlarına uygulanır.

## Komut & Sınıf Seçici

* Belli bir class değerine sahip belli bir komutu seçer.
* Örneğin, class değeri .buyuk olan paragrafları seçmek için yandaki CSS kodu kullanılabilir:

## Bitişik Kardeş (A + B) Seçici

* (p + div) : p etiketinden sonra gelen ilk div elemanını seçer.

## Genel Kardeş (A ~ B) Seçici

* (p ~ div) : p etiketinden sonra gelen tüm div elemanlarını seçer.

## Nitelik [nitelik] Seçici

* Bu seçici, etikette nitelik olup olmadığına bakıyor. Olanları seçiyor. Örneğin title niteliği olan tüm etiketleri seçmek isteseydim;
  + abbr[title] : abbr etiketlerinde niteliği title olanı seçecektir.
  + [title] : niteliği title olan tüm etiketleri seçer.

## Nitelik [nitelik=”değer”] Seçici

* Bir önceki seçiciden farklı olarak, niteliğin değerine eşit etiketleri seçmek için kullanılır. Örneğin a etiketinde target niteliği \_blank olanları seçmek istersek;
  + a[target=”\_blank”]

## Nitelik [nitelik~=”değer”] Seçici

* Nitelik içinde belirtilen değer geçen etiketleri seçmek için kullanılır.
  + <div class="box box-lg">kutu 1</div>
  + <div class="box-lg box">kutu 2</div>
  + <div class="box-lg">kutu 3</div>
* Ve class niteliğinde box geçen etiketleri seçmek için şöyle yazabiliriz;
  + [class~="box"]

## Nitelik [nitelik|=”değer”] Seçici

* Nitelik belirtilen değerle başlayan etiketleri seçer. Ancak burada önemli olan, belirtilen değerin tüm kelimeyle eşleşiyor olmasıdır. Ya tüm kelimeyle ya da tire işareti ile ayrılmış kelimenin başlangıcı ile eşleşiyor olması gerekir. Örneğin html yapımız şöyle olsun;
  + <div class="box-sm">kutu 1</div>
  + <div class="box-lg">kutu 2</div>
  + <div class="boxlg">kutu 3</div>
* Ve box ile başlayan etiketleri seçecek olsaydık;
  + [class|="box"]
* Sonuç olarak en alttaki hariç diğerlerini seçmiş olacaktık. Çünkü en alttaki başlangıç olarak tüm kelimeyi baz aldığımızda box değil boxlg olduğu için. Ayrıca boşluk ile ayrılmış nitelik değerlerinde de bu seçici işe yaramaz. Yani başlangıcı seçtiğimiz değer bile olsa boşluk varsa o etiketi seçmeyecektir.

## Nitelik [nitelik^=”değer”] Seçici

* Yukarıdaki seçicinin aksine belirtilen değer ile başlayan tüm etiketleri seçmek için kullanılır. Boşluk tire işareti vs. hiçbirine bakmaz. Tek baktığı, niteliğin değeri belirlenen değerle başlıyor mu başlamıyor mu? Yine aynı yapıda bir html yapımız olsaydı;
  + <div class="box-sm">kutu 1</div>
  + <div class="box-lg">kutu 2</div>
  + <div class="boxlg">kutu 3</div>
* Ve şu şekilde seçme işlemi yapsaydık;
  + [class^="box"]
* Sonuç, hepsini seçecektir.

## Nitelik [nitelik$=”değer”] Seçici

* Bir üstteki seçicinin tam tersi olarak, niteliği belirtilen değerle biten tüm etiketleri seçmemizi sağlıyor. Örneğin html yapımız şöyle olsun;
  + <div class="box-lg box">kutu 1</div>
  + <div class="bbox">kutu 2</div>
  + <div class="lg-box">kutu 3</div>
  + <div class="kutu">kutu 4</div>
* Şimdi class niteliği box ile biten etiketleri seçelim.
  + [class$="box"]
* Sonuç ise, son div hariç hepsini seçecektir.

## Nitelik [nitelik\*=”değer”] Seçici

* Sonu ve başlangıcı belirlediğimiz değer ile bitenleri seçmeyi başardık. Bu seçici ile, niteliklerde belirlediğimiz değer geçen tüm etiketleri seçmek için kullanılıyor.
  + <div class="box">kutu 1</div>
  + <div class="tox">kutu 2</div>
  + <div class="hoxxx">kutu 3</div>
* Şimdi class niteliği içerisinde ox değeri geçen etiketleri seçelim.
  + [class\*="ox"]

## Sözde Öğeler (Pseudo-elements)

* Bir elemanın belli bir kısmını seçmeye yarayan seçicilerdir. Örneğin, bir tablonun ilk satırı, bir paragrafın ilk harfi veya tablonun tek satırları seçilebilir. Sıklıkla kullanılan kısmi seçiciler aşağıdaki gibidir:

### ::after Sözde Öğesi

* ::after sözde elemanı ile, seçilen etiketin sonrasına bir değer ekler ve stil işlemi uygulamamızı sağlar. Bu sözde eleman ile birlikte content özelliği kullanılır, content özelliği olmadan bu sözde elemanın bir etkisi olmaz.

Örnek verecek olursak;



### ::before Sözde Öğesi

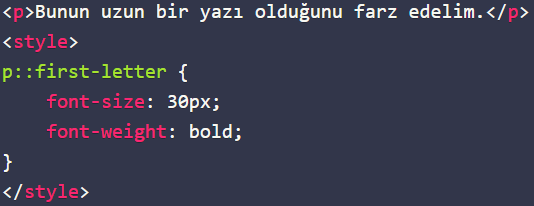
* ::before sözde elemanı ile, seçilen etiketin öncesine bir değer ekler ve stil işlemi uygulamamızı sağlar. Bu sözde eleman ile birlikte content özelliği kullanılır, content özelliği olmadan bu sözde elemanın bir etkisi olmaz.

Örnek verecek olursak;



### ::first-letter Sözde Öğesi

* Sıkça haber sitelerinde ilk harfin daha büyük boyutta ya da renkte olduğunu görürüz. İşte ilk harfi seçerek stil işlemi uygulamak için bu sözde elemanı kullanıyoruz.

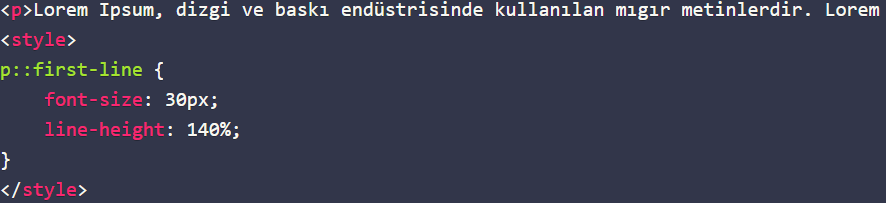


Sonuç;

**B**unun uzun bir yazı olduğunu farz edelim.

### ::first-line Sözde Öğesi

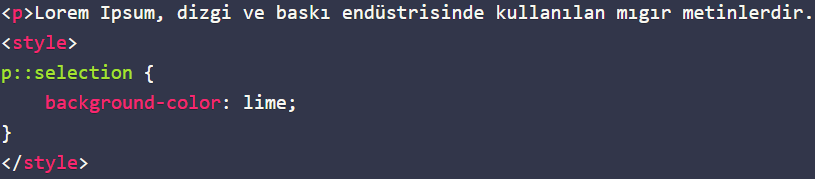
* Seçilen etikette ilk satırı seçerek işlem yapmak için bu sözde eleman kullanılır. İlk satır uygulanan stil işlemine göre farklılık gösterebilir, çünkü her zaman görünen ilk satır ne ise ona stil işlemi uygulanır. Bazen yazının boyutunu büyüttüğümüzde görünen ilk satır daha az kelimeden oluşabilir.



### ::selection Sözde Öğesi

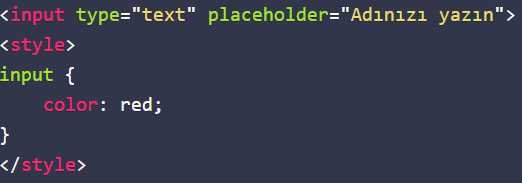
* Seçilen yazılara stil işlemi uygulamak için bu sözde eleman kullanılır. Genelde işletim sistemine göre farklılık gösteren bu seçme işleminin stillerini kendinize göre ayarlayabilirsiniz.

Örneğin aşağıdaki yazıyı seçtiğinizde varsayılan olarak ne ise o şekilde bir stil göreceksiniz. Muhtemelen windows'tan bakıyorsanız arkaplanı mavi yazı rengi beyaz olacak seçtiğiniz yerlerin.

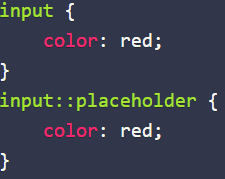


### ::placeholder Sözde Öğesi

* Form etiketlerinde placeholder niteliği bulunan etiketlere stil işlemi uygulamak için bu sözde eleman kullanılır. Şöyle ki;



* Bu örnekte “Adınızı yazın” değeri kırmızı olmayacaktır, ancak input'un içerisine bir şeyler yazarsanız bunun kırmızı renkte olduğunu görürsünüz. Yani placeholder değerine stil işlemi vermek için bu sözde öğeyi kullanabiliriz.



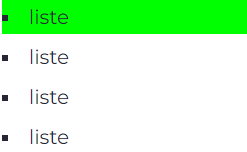
## Sözde Sınıflar (Pseudo-class)

### :first-child Sözde Sınıfı

* İlk çocuk etiketi bulmak için bu sözde sınıf kullanılır. Bu sözde sınıf kullanılan etiketin mutlaka bir ebeveyn etiketi olmak zorundadır. Örneğin;

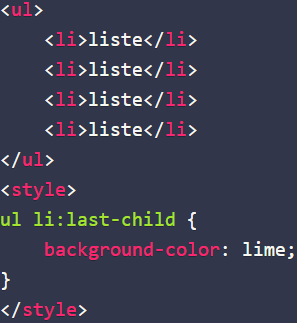


Bu kod ile ilk <li> etiketi seçilerek stil işlemi uygulanır. Çünkü li etiketinin ebeveyn etiketi <ul> etiketidir.

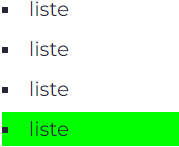


### :last-child Sözde Sınıfı

* Son çocuk etiketi bulmak için bu sözde sınıf kullanılır. Bu sözde sınıf kullanılan etiketin mutlaka bir ebeveyn etiketi olmak zorundadır. Örneğin;



Bu kod ile son <li> etiketi seçilerek stil işlemi uygulanır. Çünkü li etiketinin ebeveyn etiketi <ul> etiketidir.

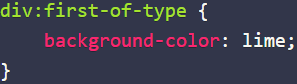


### :first-of-type Sözde Sınıfı

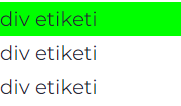
* :first-child sözde sınıfının yapamadığı şey, ilk eleman seçtiğimiz etiket değilse seçemiyordu. İşte bu durumda kullanılacak olan bu sözde sınıfıdır. Örnek vermek gerekirse;



Yukarıdaki CSS işlemi geçerli olmayacaktır çünkü ilk elemanımız div etiketi yerine p etiketidir. Fakat burada direk tipe bakarak işlem yaptırmış olsaydık, yani şöyle;



Sonuç başarılı olacaktı;



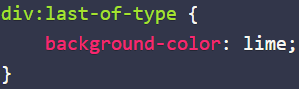
Yani burada önemli olan seçtiğimiz tipte bir etiketin olup olmadığı varsa onun ilk elemanını seçip stil işlemi uygulayabiliyoruz.

### :last-of-type Sözde Sınıfı

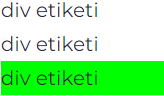
* :last-child sözde sınıfının yapamadığı şey, son eleman seçtiğimiz etiket değilse seçemiyordu. İşte bu durumda kullanılacak olan bu sözde sınıfıdır. Örnek vermek gerekirse;



Yukarıdaki CSS işlemi geçerli olmayacaktır çünkü son elemanımız div etiketi yerine p etiketidir. Fakat burada direk tipe bakarak işlem yaptırmış olsaydık, yani şöyle;

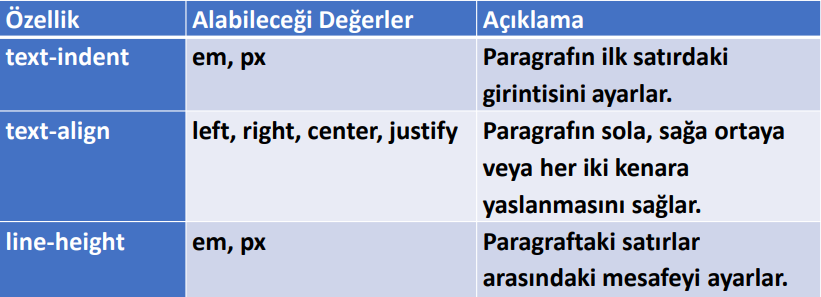


Sonuç başarılı olacaktı;



Yani burada önemli olan seçtiğimiz tipte bir etiketin olup olmadığı varsa onun ilk elemanını seçip stil işlemi uygulayabiliyoruz.

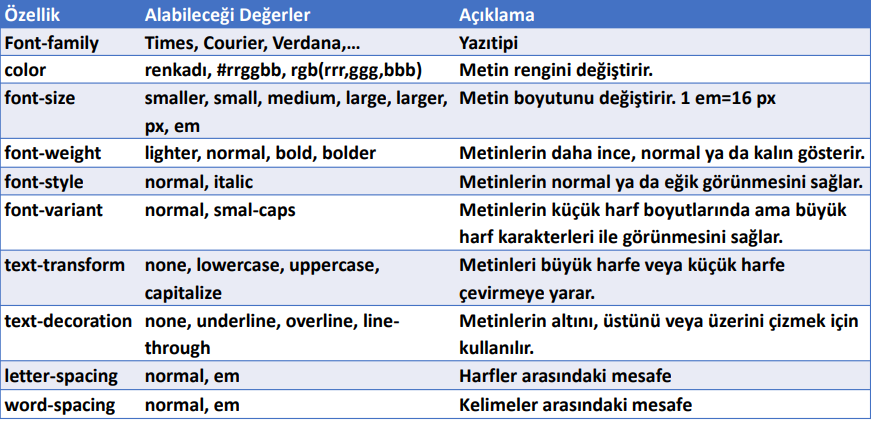
## Paragraf Özelikleri



NOT\*\*text-align özelliği inline etiketlerde çalışmaz.

* + text-align-last: paragrafın son satırı için işlem yapar.

## Yazıtipi Özellikleri

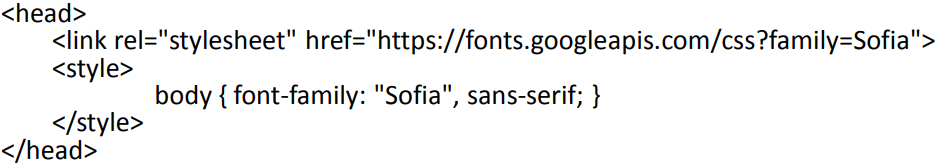


* font-size: 1em; 🡺 em body font-size değerine çalışır. Eğer kapsayıcı öğesinin bir font-size değeri var ise onu baz alacaktır. body etiketinde font-size tanımlı değil ise tarayıcılar otomatik olarak bu değeri 16px olarak kabul eder ve 1em = 16px olur. h1{font-size:2em} = 32px
* font-size: 1rem; 🡺 rem birimi sadece html öğesinin font-size ölçüsünü baz almaktadır. html{font-size:16px;} ise 1rem = 16px olacaktır.
* white-space: nowrap; 🡺Metin <br> etiketi ile karşılaşana kadar aynı satırda devam eder.
* white-space: pre; 🡺HTML'deki <pre> etiketi gibi davranır. Yalnızca kaynaktaki yeni satır etiketinde ve etiketi ile alt satıra iner.
* text-shadow: 🡺 Yazılara gölge vermemizi sağlar.
* text-shadow: yatay-mesafe dikey-mesafe yumuşaklık renk;

## Web Güvenli Yazıtipleri

* Web güvenli yazıtipleri yaygın olarak tüm cep telefonu, tablet ve bilgisayarlarda kullanılan yazıtipleridir. Bu yazıtipleri:
  + Arial, Verdana, Times, Tahoma, Courier, Helvatica gibi yazıtipleridir.
* Eğer ziyaretçi bilgisayarında sayfada kullandığımız yazıtipi yoksa tarayıcı tarafından belirlenen varsayılan yazıtipi kullanılır. Bu da sayfa tasarımının bozulmasına neden olabilir.

## Google Yazıtipi Kullanımı

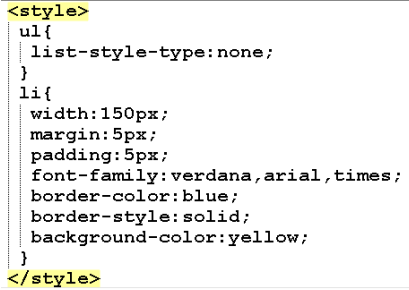
* Yaygın olarak kullanılan yazıtipleri yerine web sayfalarınızda ücretsiz olarak sağlanan Google yazıtiplerini kullanabilirsiniz.
* Google yazıtiplerini kullanmak için sayfanın head bölümüne aşağıdaki gibi bir CSS kodu yazılmalıdır:

## Metin Ardalan İşlemleri

Metinlerin ardalan işlemleri kutu modelindekiyle aynıdır.

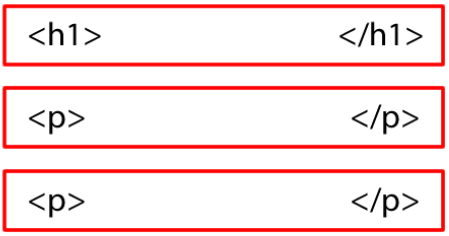
* Ardalan rengi için:
  + background-color
* Ardalan resmi için:
  + background-image

## Listelerin Biçimlendirilmesi

* Madde imlerini değiştirmek için **list-style-type** özelliği kullanılır.
  + none (madde imi yok)
  + disk (daire)
  + circle (çember)
  + square (kare)
  + decimal (numara)
  + lower-alpha (küçük harf a, b, c)
  + lower-greek (yunanca küçük harf)
  + lower-latin (latince küçük harf a, b, c,)
  + lower-roman (küçük roma rakamı i, ii, iii, iv)
  + upper-alpha (büyük harf A, B, C)
  + upper-greek (yunanca büyük harf)
  + upper-latin (latince büyük harf A, B, C)
  + upper-roman (büyük roma rakamı I, II, III, IV)
  + url(“resim”) (madde imi yerine resim görüntülenir)

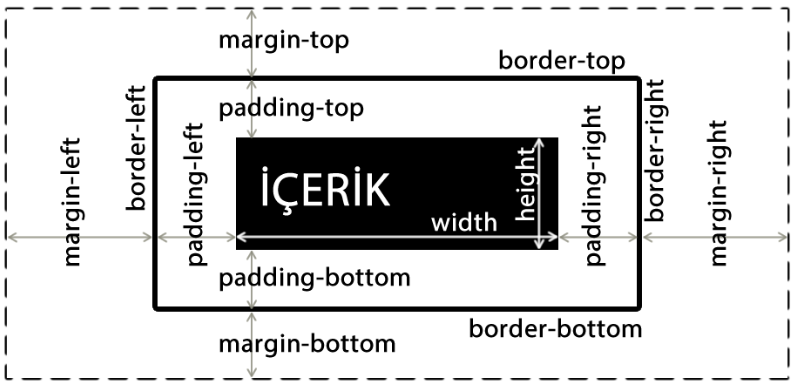
## CSS Kutu Modeli Nedir?

* CSS web sayfasında bulunan HTML komutlarını birer kutu olarak görür. Yani, aşağıdaki gibi bir şey görür.



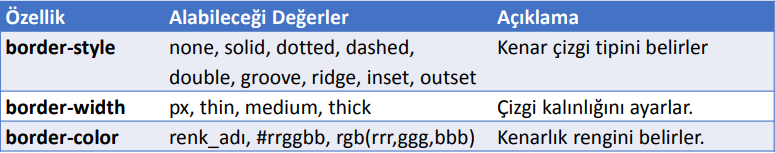
## CSS Kutu Modeli Nedir?

* CSS ’e birer kutu gibi görünen HTML komutları içerik, iç-aralık (**padding**), kenarlık ve dış-aralıktan (**margin**) oluşmaktadır. Yani, her bir HTML komutu kutu modelinde aşağıdaki parametrelere sahiptir.



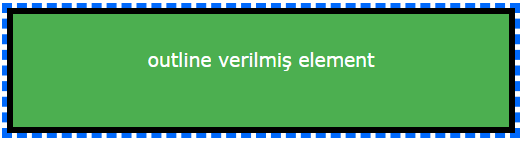
## Kutu Kenarları (border)

* Kutu kenarlıklarını kontrol etmek border özelliği kullanılır.



## Outline Özelliği(Dış Çizgi)

* CSS outline özelliği dış çizgi için renk, genişlik gibi özellikleri belirler. outline denen dış çizgi border yani sınırın dışında tüm elementi saran bir çizgidir. outline özelliği border özelliğinden farklıdır. outline özelliği elementin genişliğinin ve yüksekliğinin bir parçacı değildir. outline belirlediğinizde elementin genişliği ve yüksekliği değişmez.

Örnekte siyah çizgi border, hemen onun dışındaki kesik mavi çizgi outline tarafından çizilmiştir.

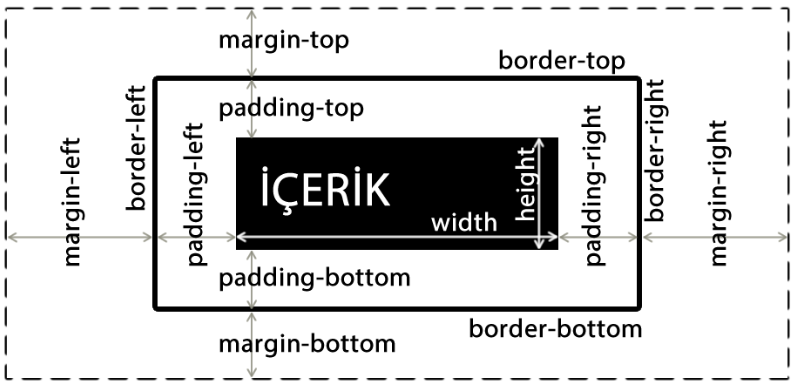
## Kutu Boyutu ve Aralıklar

* Kutu boyutunu belirlemek için width ve height özellikleri kullanılır.

NOT\*\*İnline etiketler width ve height özelliğini kullanamazlar.

* max-height özelliği, bir etikete maksimum yükseklik değeri vermek için kullanılır.
* max-width özelliği, bir etikete maksimum genişlik değeri vermek için kullanılır.
* min-height özelliği, bir etikete minimum yükseklik değeri vermek için kullanılır.
* min-width özelliği, bir etikete minimum genişlik değeri vermek için kullanılır.
* max-content değeri, içeriğinin genişliği kadar yer kaplar.
* min-content değeri, tek tek kelimelere bakar. En uzun kelimenin genişliği kadar üst üste bütün kelimeleri dizer.

Bu özellikleri daha çok responsive tasarımlarda kullanırız.



## Kutu İç Aralık (Padding) Ayarı

* İçerik ile kenarlık arasındaki mesafe padding özelliği ile ayarlanır.

## Kutu Dış Aralık (Margin) Ayarı

* Kutu kenarı ile diğer nesneler arasındaki mesafeyi belirlemek için margin özelliği kullanılır.

NOT\*\*İnline etiketler üstten ve alttan margin özelliğini kullanamazlar.

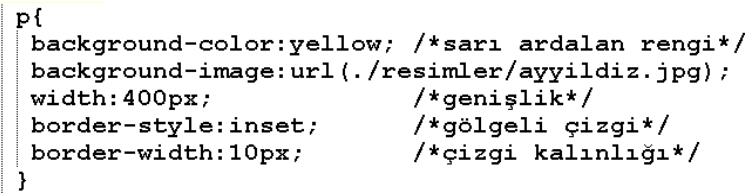
## Kutu Boyutunun Sabitlenmesi

* Fark ettiğiniz üzere margin, border ve padding komutları ile verilen değerler **içerik** boyutuna eklenmektedir.
* Örneğin, resim 200px genişlikte ise, border-width:5px ve padding:10px komutundan sonra nesne genişliği (5px+10px+200px+5px+10px) 230px olur.
* Bu tür nesnelerin boyutunu sabit tutmak için box-sizing özelliği kullanılır:



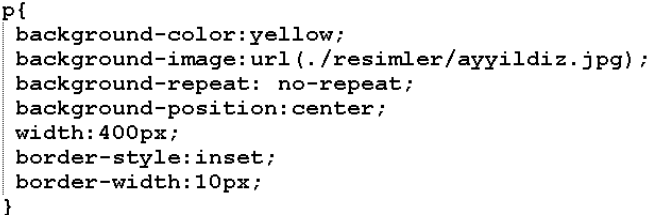
## Kutu Arka Plan Rengi & Resmi

* Kutuların arka planını değiştirmek için background özelliği kullanılır. Ayrıca, aşağıdaki özellikleri de kullanabilirsiniz:
  + backgroud-color: Arka plan rengini ayarlar.
  + background-image: Arka plana resim koymak için kullanılır.



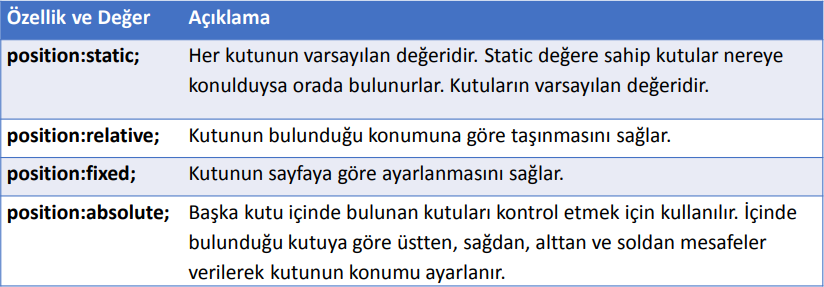
## Kutu Arka Plan Resim Ayarları

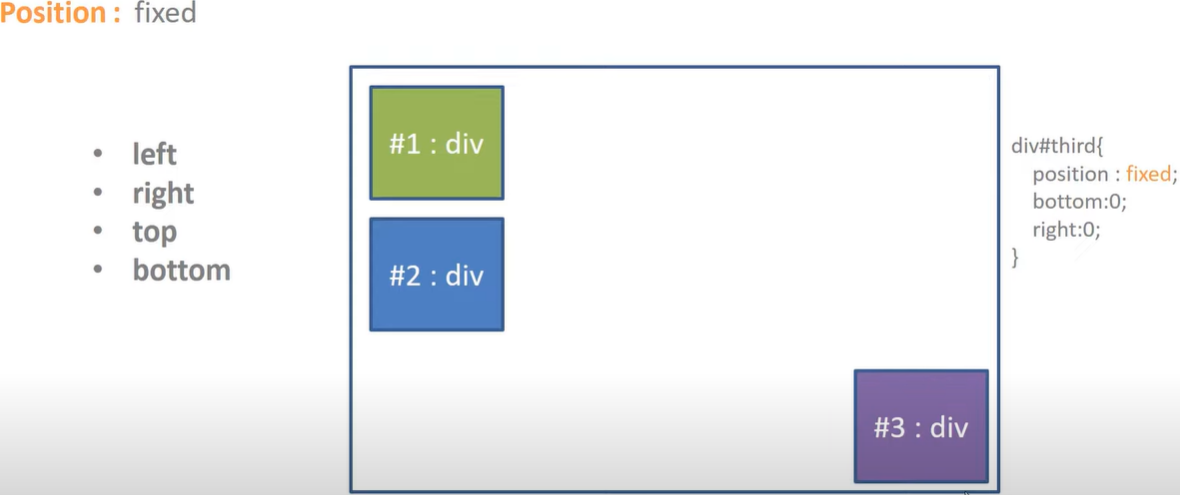
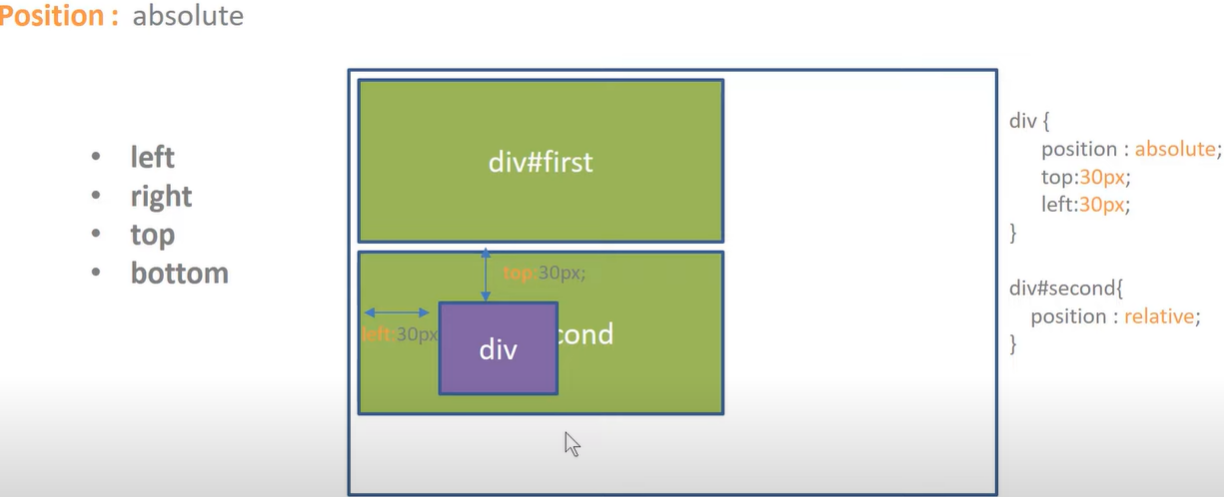
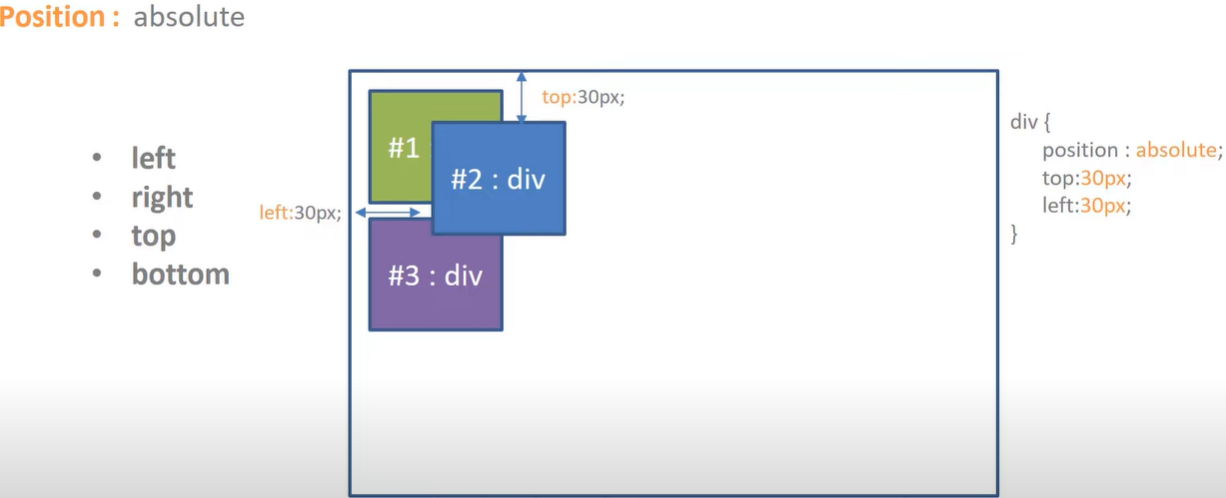
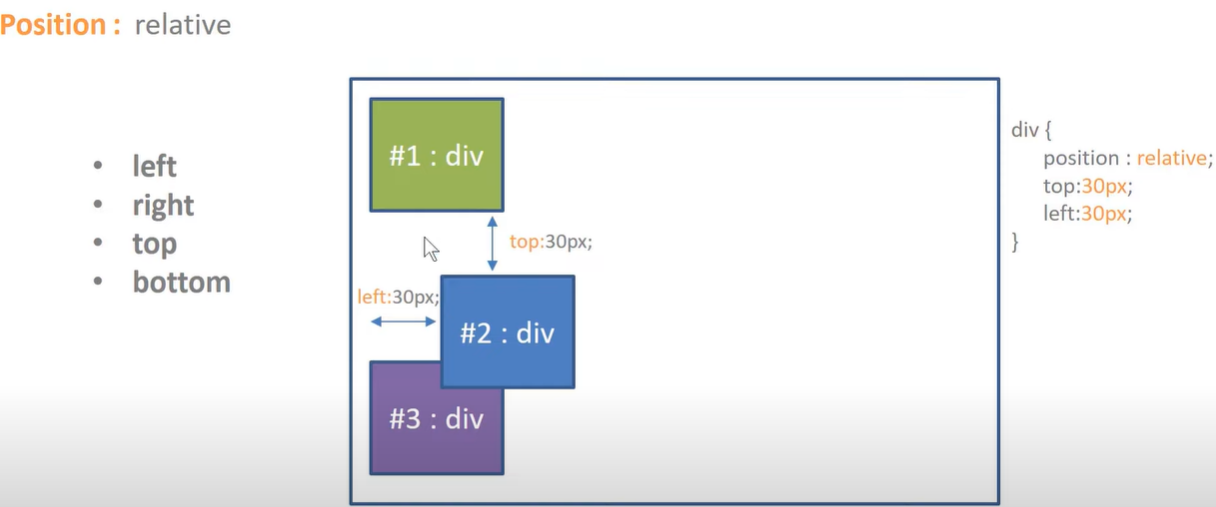
* Arka plan resminin konumunu belirlemek için aşağıdaki özellikler kullanılır:
  + background-size: Arka plan resminin boyutunu değiştirir.
  + background-repeat: Arka plan resminin tekrarlanıp tekrarlanmayacağını belirtir.
  + background-position: Arka plan resminin konumunu ayarlar.
  + background-attachment: Arka plan görselinin ekranda kaydırılıp kaydırılmayacağını veya sabitleneceğini belirtir.
  + background-clip: Arkaplan görseli ya da arkaplan renginin başlangıç konumunu belirler.
  + background-origin: Arkaplan görselinin başlangıç konumunu belirler.



## Kutunun Sayfaya Konumlandırılması (position)

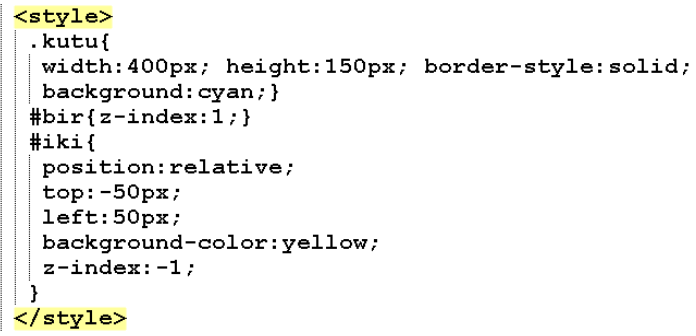
* Varsayılan olarak HTML komutları yukarıdan aşağı doğru sayfa soluna hizalanmış olarak yerleştirilir. Kutuların sayfada bulundukları yeri değiştirmek için position özelliği kullanılır.
  + position: sticky; Kaydırma çubuklarını takip eder ve istenilen pozisyon yakalandığında nesneyi o noktada sabitler.





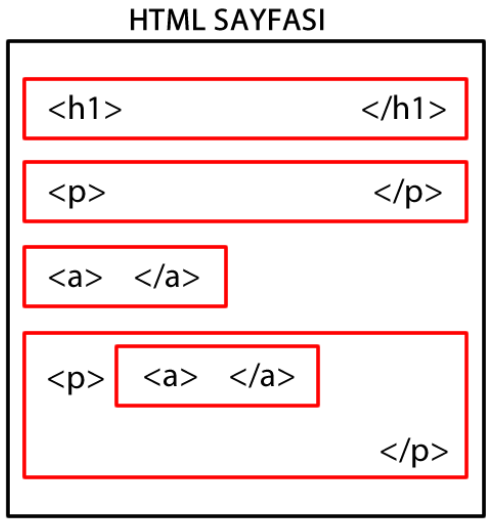
## Kutuların Öne-Arkaya Alınması (z-index)

Kutuların öne veya arkaya taşınması için z-index özelliği kullanılır. Daha yüksek z-index değerine sahip olan daha önde görünür.



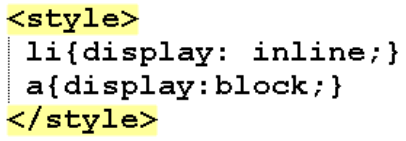
## Block ve Inline Komutlar

* Kutu modelinde bazı HTML komutları block, bazıları HTML ise inline olarak çalışır. Blok olarak çalışan hx, table veya p gibi komutlar sayfanın solundan sağına kadar bulundukları tüm satırı kaplarken, inline olarak çalışan a, b, em veya img gibi komutlar sadece bulundukları alanı kaplar.



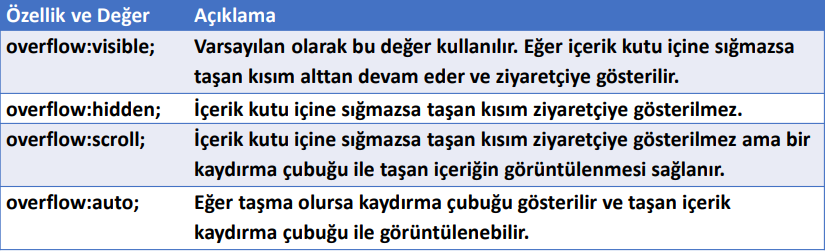
## Kutunun Davranışı (display)

hx, p, li, form, table veya div gibi komutlar block davranışına sahiptir ve bulundukları yerdeki tüm satırı işgal ederler. a, span, em, b veya i gibi komutlar ise inline davranışa sahiptir ve sadece bulundukları yeri işgal ederler. Bazı durumlarda inline komutların block, block komutların ise inline şeklinde davranmasını isteyebiliriz.



## Kutudan Taşan İçeriğinin Ayarlanması (overflow)

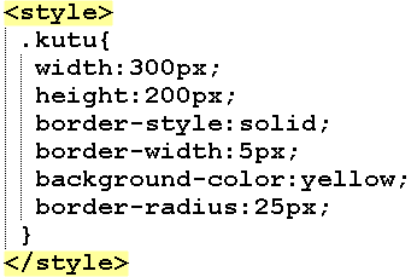
Eğer küçük bir kutunuz varsa ve içine alabileceğinden daha fazla içerik koyarsanız adına overflow/taşma denilen olay meydana gelir. overflow özelliği ile taşan içeriğin durumu belirlenebilir.



* + hyphens: auto; 🡺 Satır atlama çizgileri otomatik olarak oluşturuluyor.

## Kutu Köşelerinin Yuvarlatılması (border-radius)

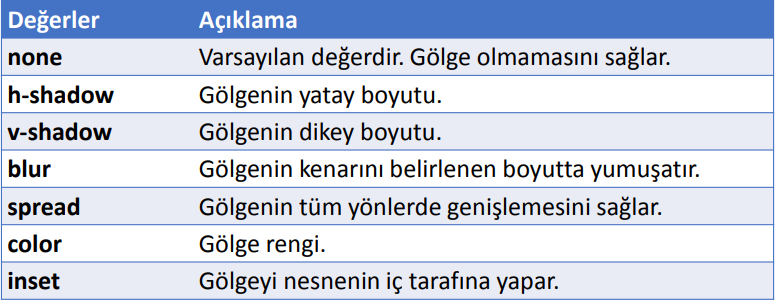
Kutu kenarlarını yuvarlatmak için border-radius özelliği kullanılır.



## Gölge Oluşturma (box-shadow)

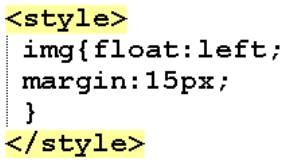
Kutuya gölge yapmak için box-shadow özelliği kullanılır.

* + box-shadow: h-shadow v-shadow blur spread color inset;



## Kutuların Yaslanması (float)

Kutunun sola veya sağa yaslanması için float özelliği kullanılır. Diğer kutular float özelliği ile yaslanan kutunun etrafını sararlar. Eğer nesnelerin kutu etrafını sarmasını istemiyorsanız clear özelliğini kullanırsınız.



## Geçiş Efektleri (transitions)

CSS transition Özelliği bir öğe hover olduğunda ya da .active gibi bir classname aldığında değişen stil özelliklerini öğeye geçişli bir şekilde vermeyi sağlar.

* + transition-property
  + transition-duration
  + transition-timing-function
  + transition-delay
  + transition kısa yolu yazımı.

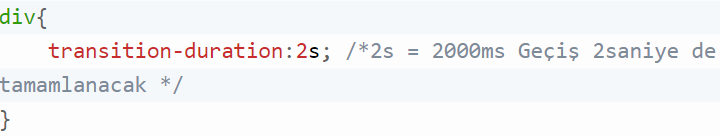
## Transitions – property Özelliği

* transition-property ile geçiş işlemleri hangi css özelliklerine uygulanması gerektiğini belirtir.
* Tüm CSS özelliklerine uygulanması isteniyorsa, color, background-color, margin, padding, vb. transition-property:all; şeklinde kullanılabilir.
* Birden fazla transition-property değeri girmek için her değer arasına (,) virgül koyunuz.



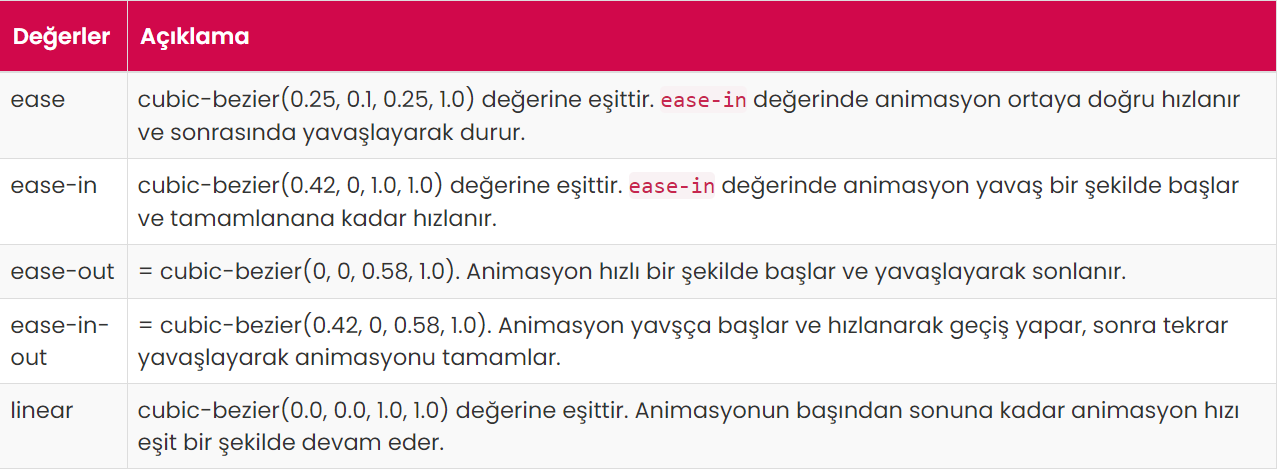
## Transitions – duration (Geçiş Süresi) Özelliği

* transition-duration geçiş süresini aynı animation-duration da belirttiğiniz gibi saniye ve milisaniye cinsinden bir süre girmeniz gerekir.



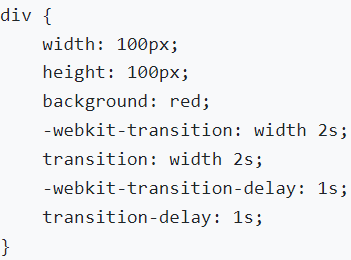
## Transitions – timing – function (Geçiş Zamanı Yapılandırması)

* Bu işlev, bir öğenin bir durumdan diğerine geçişini yumuşatmak veya hızlandırmak için kullanılır. Özellikle CSS geçişleri ve animasyonlarıyla ilgili belirli bir geçiş efekti oluşturmanıza yardımcı olur.

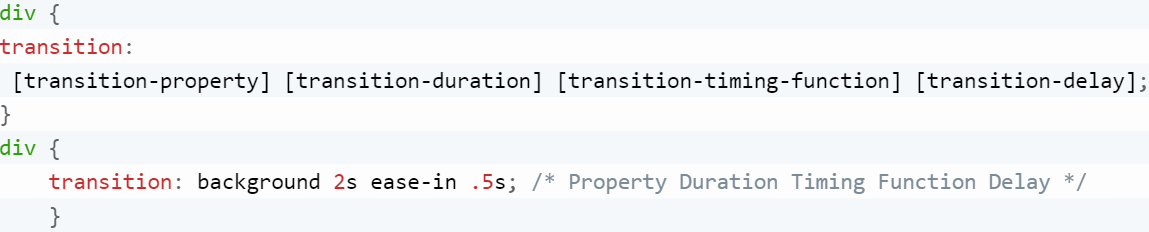


## Transitions – delay (Geçişi Geciktirme)

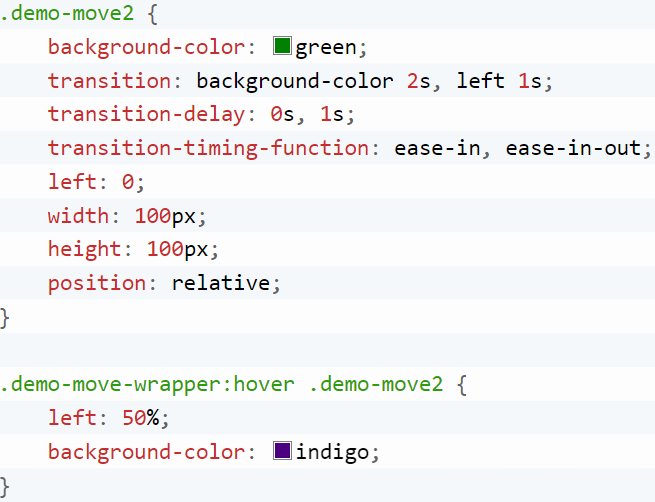
* transition-delay css özelliği çok sık kullanılan bir transition özelliğidir. Örnek olarak sahneye kapsayıcı div elemanına: hover yaptığınızda 2 adet öğe getirmek istiyorsunuz. 1 öğeyi hemen ve 2. öğeyi de .5s gecikmeli olarak getirmek istiyorsunuz. İşte bu noktada 2. öğeye transition-delay özelliğini ekleyiniz.



## Transitions CSS Kısa Yolu



## Transitions CSS Örneği



## Tasarımda Kullanılan Renk Düzenleri

* CSS kodlarında renkleri belirtmek için aşağıdaki yöntemler kullanılabilir:
  + Renk adları (red, green, blue, vb…)
  + HEX (Onaltılık) renk kodları
  + RGB değerleri
  + HSL değerleri
* Renkler arası yumuşak bir geçiş sağlamak için gradient kullanılır.

## Saydamlık

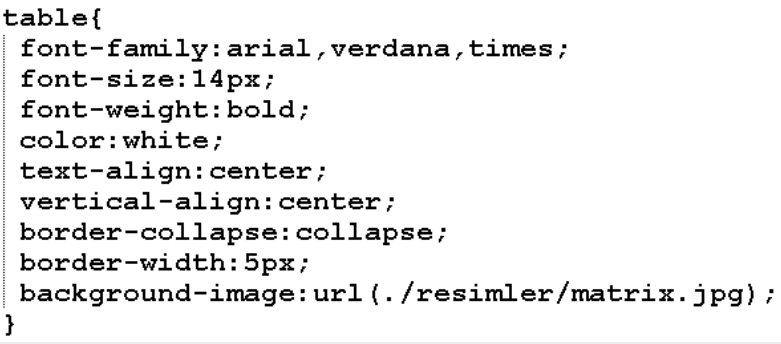
• Nesneleri saydam veya opak yapmak için opacity komutu kullanılır.

## Tablo Biçimlendirme

Tablo biçimlendirmede table, tr, td, th, caption, thead, tbody ve tfoot etiketlerine ayrı ayrı CSS kodları uygulanabilir.

* Hücre içindeki metinleri hizalamak için aşağıdaki özellikler kullanılır:
  + text-align: Bir metnin yatay hizalamasını ayarlamak için kullanılır.
  + vertical-align: Bir metnin dikey hizalamasını ayarlamak için kullanılır.
* Eğer iki-çizgi şeklinde görünen kenarlıkları tek çizgiye dönüştürmek istiyorsanız border – collapse : collapse; komutunu kullanmalısınız.
* Tablo veya hücrelere ardalan için:
  + background-color
  + background-image
  + empty-cells: hide; tablodaki boş hücreleri gizler

## Tablo Biçimlendirme Uygulaması



## Formların Biçimlendirilmesi

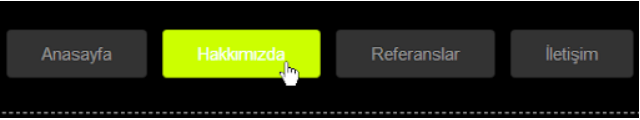
* Form elemanları kutular gibi biçimlendirilir.
* Form elemanın içi ve kenarları biçimlendirilebilir.
* Form elemanlarının içindeki metinler biçimlendirilebilir.
* Formlardaki onay ve iptal butonları biçimlendirilebilir.

## Linklerin Biçimlendirilmesi

* Linkler metinler gibi biçimlendirilebilir.
* Linklerin farklı durumları için aşağıdaki özellikler kullanılır:
  + a:link (ziyaret edilmemiş link)
  + a:visited (ziyaret edilmiş link)
  + a:hover (fare link üzerinde)
  + a:active (fare tıkladığında)
  + transition: özelliği ile linklere kaç saniyede değişimi olacağını gösteririz.

## Menü Oluşturma ve Biçimlendirme

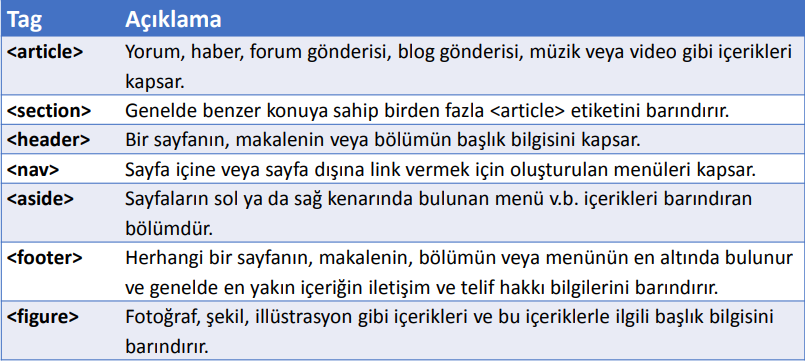
* Menüler listelerden oluşturulur.
* Menülerin biçimlendirilmesi CSS Kutu Modeli’nde öğrendiğimiz komutlarla yapılır.



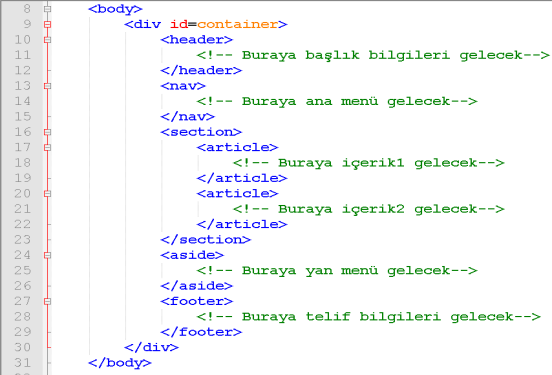
## Sayfa Düzeninin Belirlenmesi

* Çok farklı sayfa tasarımları olmasına rağmen klasik bir sayfanın genel yapısı çoğunlukla yandaki gibidir.
* Bu tasarım ve etiketlerle arama motoruna sayfayla ilgili en doğru bilgi verilmiş olur.

## Semantik Sayfa Düzeni Etiketleri

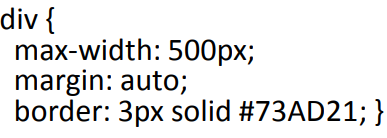


## HTML ile Sayfa Düzeninin Oluşturulması



## CSS ile Sayfa Düzeninin Biçimlendirilmesi

* Web sayfalarının küçük ekranlarda da düzgün görünebilmesi için tüm içeriklerin kutu boyutu değişebilen genişlikte ayarlanabilir. Bunun için width yerine max-width özelliği kullanılır.
* Kutuyu sayfaya ortalamak için margin: auto; komutu kullanılır.



## Hazır Şablonlar

* Bireysel veya ticari web sayfaları tasarlamak için internetten bulabileceğiniz hazır şablonları kullanabilirsiniz.
* Arama motoruna «free html template» yazarak birçok ücretsiz şablona ulaşabilirsiniz.
  + https://www.free-css.com/
  + https://html5-templates.com/
  + https://themeforest.net/

## Tepkisel Sayfa Nedir?

* Kullanılan cihaza göre boyut ve görünüm değiştiren sayfalara tepkisel / responsive sayfa denir.
* Bu özellik sayesinde sayfalar cep telefonu, tablet veya PC’de düzgün bir şekilde görüntülenebilir.

## Görüş Alanı (Viewport)

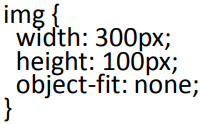
* Web sayfamızın görünen alanına görüş alanı / viewport denir.
* Eskiden web sayfaları PC ekranına göre tasarlanırdı, bu nedenle sayfalar cep telefonu ve tabletlerde kötü görünebiliyordu.
* HTML5 ile birlikte viewport özelliği geldi. Bu özellik sayesinde web sayfası ekran boyutuna tepki verebiliyor:
  + <meta name="viewport" content="width=device-width, initial-scale=1.0">

## CSS Ölçü Birimleri

* Sayfa tasarımında sıklıkla kullanılan ölçü birimleri aşağıdaki gibidir: • Mutlak ölçü birimleri:
  + px (piksel)
  + pt (punto)
* Göreceli ölçü birimleri:
  + % (yüzde, içinde bulunduğu nesneye göre)
  + em (oran, HTML komutunun font boyutuna göre)

## Nesne Sığdırma

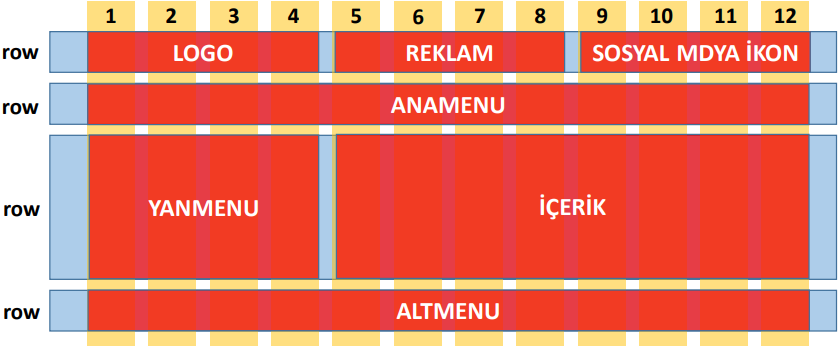
* Resim, video vb. elemanların belirlenen ölçü içine ne şekilde yerleştirileceğini belirlemek için object-fit özelliği kullanılır.



* Bu özelliğin aldığı değerler aşağıdaki gibidir:
  + none: Boyut değiştirmez.
  + fill: Varsayılan değer. İçinde bulunduğu nesnenin boyutunu alır; Genişlik ve yükseklik kutuya sığacak şekilde değişir.
  + contain: En-boy oranı korunarak içinde bulunduğu nesneye göre boyut değiştirir.
  + cover: En-boy oranı korunarak içinde bulunduğu nesneye göre boyut değiştirir, kutu içine sığmak için resim kırpılır.

## Izgara Tabanlı Sayfa Tasarımı (Grid View)

* Günümüzün modern web sitelerinde **Izgara Tabanlı Tasarım** kullanılır.
* Bu tasarım yönteminde sayfa sanal satırlardan (row) ve 12 sütündan (column) oluşur. Her kutu belli bir sütun genişliğinde tasarlanır.

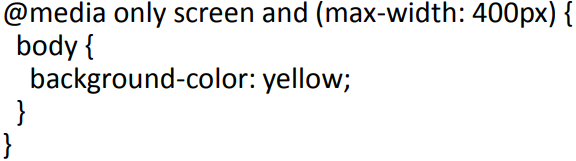


* 12 sütundan oluşan web sayfalarındaki kutular çalıştıkları cihaza göre boyut değiştirirler. Böylece sayfalar tüm cihazlarda uygun bir şekilde görünürler.

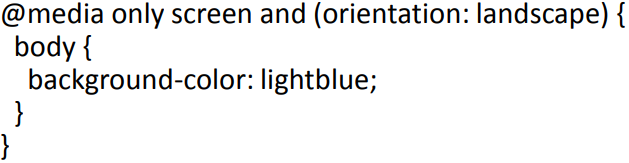


## CSS Kod Sorgusu (Media Query)

* Belli şartlar sağlandığında belli CSS kodlarını çalıştırmak için @media komutu kullanılır.
* Tarayıcı max. 400px ise (400px’den küçükse) arka plan rengi sarı olsun:



* Cihaz yatay tutulursa arkaplan gri olsun:



## Izgara Tabanlı Sayfa Uygulaması

* Günümüz web sitelerini tasarlayabilmek veya anlayıp değiştirebilmek için Izgara Tabanlı Tepkisel Sayfa tasarım mantığı iyi anlaşılmalıdır.
* İş hayatında sıfırdan sayfa tasarımı yapmak yerine, daha çok W3.CSS veya Bootstrap gibi hazır CSS kütüphaneleri kullanılır. Bu sayede daha hızlı ve daha profesyonel sayfa tasarımı yapılmış olur.
* Bakınız: https://www.w3schools.com/css/css\_rwd\_frameworks.asp
* Biz Tepkisel Sayfa konusunu temelden öğrenmeye çalıştığımız için hazır kütüphane yerine sıfırdan CSS kodu yazarak adım adım tepkisel sayfaların nasıl tasarlandığını öğreneceğiz.

## Adım 1: HTML Sayfasının Kodlanması

* Tepkisel bir sayfa oluşturabilmek için, sayfamızın head bölümüne önceki konularda öğrendiğimiz viewport komutunu koyuyoruz.

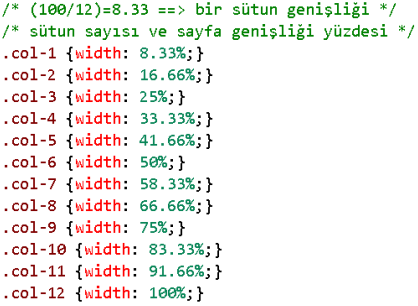


## Adım 2: Kutu Genişliğinin Sabitlenmesi

* Kod takibinin daha kolay olması için CSS kodlarını sayfamızın head bölümüne yazacağız.

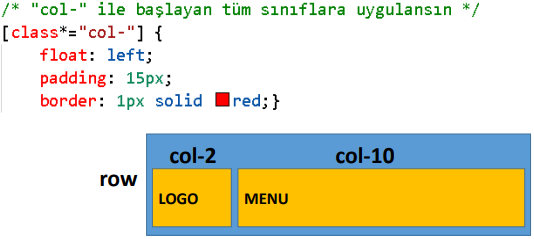


## Adım 3: Sütun CSS Kodlarının Yazılması



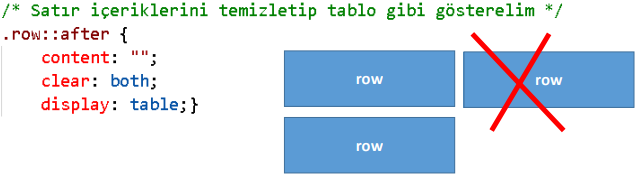
## Adım 4: Sütunların Sola Yaslandırılması

* Aynı satır içinde bulunan kutuların sola yaslandırılması ve yan yana gelebilmesi için aşağıdaki CSS kodu yazılabilir:



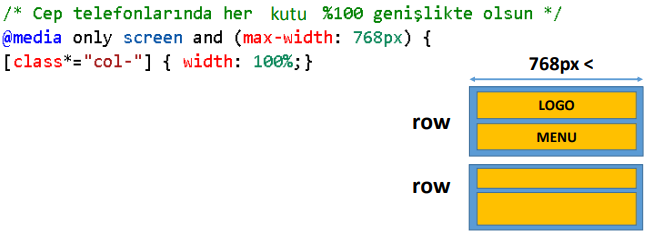
## Adım 5: Satırların Temizlenmesi

* Izgarada (gridview) kullanılan satır (row) içeriklerinin temizlenmesi, her satırın alt alta gelmesi ve float komutundan etkilenip sola yaslanmaması için aşağıdaki kodlar yazılır.



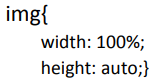
## Adım 6: Küçük Ekranlarla Uyum

* Max. genişliği 768px olan (yani, 768px’den daha küçük) ekranlarda her kutu satırın %100’ünü kaplasın.



## Adım 7: Tepkisel Resim

* Tepkisel resim oluşturmanın en kolay yolu, resmin bulunduğu kutunun içine sığmasını sağlamaktır. Bunun için aşağıdaki kodları yazmak yeterlidir:



## Adım 8: Satır ve Kutuların Oluşturulması

* CSS kodları yazıldıktan sonra body etiketi içine satır, kutu ve içerikleri oluşturan HTML kodları yazılır.

