**2. CSS**

**Css’te Değişken Tanımlamak**

Daha önce herhangi bir preprocessor kullanmışsanız eğer syntax size garip gelebilir.

Değişken adlarının başına *çift çizgi*koyarak adlandırmaya başlıyorsunuz.

**/\* CSS \*/**

 --color: red;

⚠️ Burada bir uyarı vermek istiyorum. Normalde css propertylerinde case-sensitive olayı yoktur. Fakat değişken adlarında bu geçerli değil. *Yani değişken adları case-sensitive’dir.*

️--color ile --Color aynı şey değildir!

Değişkene erişmek için ise *var* keywordu kullanılır.

*var( — degisken-adi, deger)* kullanımı da mevcuttur.

**/\* CSS \*/**

--color: red;

--div-font: "Georgia";div{

   background-color: var(--color);

   font-family: var(--div-font, "Arial", serif); 

                // font-family: "Georgia", "Arial", serif;

}

**Değişken Scope**

Değişken tanımlandığı yerlerde geçerlidir.

*:root{ }* da tanımlanan bir değişken tüm html’de geçerliyken, *#item{ }* da tanımlanan bir değişken sadece *#item*’da geçerlidir. Ona dışarıdan erişilemez.

**/\* CSS \*/**

:root{

  --color: red;

}#item{

   --color: blue;

}p{

   --color: black;

}**<!-- HTML -->**

<html>

 <body>

    <div> Bu div'in rengi kırmızıdır!</div>

    <div id="item">

       Bu div'in rengi mavidir!

       <p> Bu p'nin rengi mavidir! Çünkü parent'ı #item'dır.</p>

    </div>

    <p>Bu p'nin rengi siyahtır!</p>

 </body>

</html>

Farklı scopelarda aynı değişken adını farklı değerlerde kullanabilirsiniz.*— color*değişkeni *:root*, *#item*, *p* ‘de farklı değerler aldığını yukarıda görüyorsunuz.

**Yanında Birim Olan Propertylerde Kullanım**

“px” gibi birimlerle kullanılan değerlerde kullanım biraz farklı.

**/\* CSS \*/**

:root{

   --space: 10;

}.box{

   padding: var(--space)px;

}

❌ Bu kullanım geçerli değildir..box{

   padding: calc(var(--space) \* 1px);

}

✔️ calc() fonksiyonu kullanarak geçerli hale getirilir.

**CSS DİSPLAY KULLANIMI**

**Display** özelliği bir HTML elemanı için kullanılan kutu türünü belirtir. Web sayfamızda elemanların ne tür konumlandırılacağa ve sayfamızda görünüp , görünmeyeceğini belirler. Genel bir tanımlamayla bu özellik, elamanın hangi eleman gibi davranacağını belirtir.

Web sayfalarımızı oluşturan HTML etiketleri, sayfaya yerleşimlerine göre **satır seviyesi** elemanlar ve **blok seviyesi** olmak üzere iki gruba ayrılırlar. web sayfa tasarımında bu iki özelliği anlamak önemlidir.

**(inline) Satır seviyesi elemanlar**

Bu gruptaki **HTML** etiketleri, yeni bir satır başlatmazlar. **İnline** adı üzerinde aynı hizada, yan yana görüntülenirler yeni bir satır başlatmazlar. Yükseklik ve genişlik gibi özellikler verilemez. Bu elemanlar kendisini oluşturan içerik kadar bir alan kaplarlar ve satır sonunu zorlamaz yani bir alt satıra geçmezler. örneğin Anasayfa şeklinde bir yazıya link vermişsek bu elemanın kaplayacağı alan yazının kendisi kadardır. Satır seviyesi elemanlara örnek verirsek:

·       **span**

·       **a**

·       **b**

·       **input**

·       **img**

·       **Label**

·       **strong**

·       **small**

Satır elemanlarını örnekte gösterelim:

CSS:

span,b{

   color:white;

   background-color:gray;

   width:200px;

   height:100px;

  }

HTML:

<p>Uğraşma boşuna. <b>Seni</b> ancak <span> gördükleri ve duydukları</span> kadar anlayacaklar.</p>

ÇIKTI:



Yukarıda çıktıda görüldüğü gibi **<b>** etiketi ve **<span>** etiketi satır elemanı olduğu için satır içinde yan yana görüntülenmeye devam etmişlerdir. **b** ve **span** etiketlerine **yükseklik** ve **genişlik** verdiğimiz halde uygulanmamıştır. Satır içi elemanlara genişlik ve yükseklik uygulanmaz.

**Satır** elemanını **blok** seviyesi elemana dönüştürmeye örnek verelim.

HTML:

<div id="header">

    <a href="#">Anasayfa</a>

    <a href="#">Hakkımızda</a>

    <a href="#">İletişim</a>

</div>

ÇIKTI:

[Anasayfa](https://www.webcebir.com/129-css-display-kullanimi-dersi.html) [Hakkımızda](https://www.webcebir.com/129-css-display-kullanimi-dersi.html) [İletişim](https://www.webcebir.com/129-css-display-kullanimi-dersi.html)

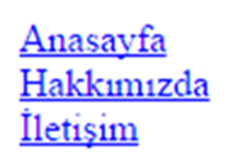
Yukarıda çıktıda görüldüğü üzere Header alanındaki bu üç **link** yan yana görüntülenirler. Çünkü **linkler satır seviyesi** elemanlardır ve display özellikleri varsayılan olarak **inline** değerini alır. Bu linkleri alt alta görüntülemek istersek display özelliğini **block** yapmaktır. Bunu sağlamak için aşağıdaki CSS kuralını CSS dosyamıza eklememiz gerekir.

.header a{

    display:block;

}

ÇIKTI:



**Blok seviyesi elemanlar**

Bu gruptaki **HTML** etiketleri ise eklendikleri zaman yeni bir satıra yerleşirler; Öncesi ve sonrası satır sonu olan bir elementtir. Bulundukları yerin mevcut tam genişliğe sahip olurlar. **Blok** seviyesi elemanların **yüksekliği, genişliği, dolgu, kenar boşlukları** vb. özellikleri ayarlanabilir. Blok seviyesi elemanlara örnek verirsek:

·       **H1**

·       **P**

·       **Li**

·       **Div**

·       **UL**

·       **Table**

Block seviyesi elemanlarına örnekte gösterelim:

HTML:

<ul type="disc">

  <li> Anasayfa </li>

  <li> Hakkımızda </li>

  <li> Adresimiz </li>

  <li> İletişim </li>

</ul>

ÇIKTI:

·        Anasayfa

·        Hakkımızda

·        Adresimiz

·        İletişim

Yukarıda çıktıda gördüğümüz gibi liste elemanları blok elemanıdır. Blok elemanları her sayfaya eklediğimizde alt alta sıralanır.

Her **HTML** elemanı varsayılan modu vardır, örneğin, <span> etiketi Bir satır içi eleman varsayılı olarak gelir. <h1> etiketi varsayılı olarak bir blok elemanıdır. Buna rağmen, her zaman hemen hemen herhangi bir **HTML** elemanı ihtiyaçları karşılamak için değiştirilebildiğini unutmayın. **Display** özelliği ile **inline(satır)** elemanları bir **blok** elemanı haline dönüştürülebilir. Bir **blok** elemanıda **inline(satır)** elemanı haline dönüştürülebilir.

**DİSPLAY ÖZELLİĞİNİN ALACAĞI DEĞERLER**

|  |  |
| --- | --- |
| **DEĞER** | **TANIM** |
| **none** | Bu eleman görünmez. |
| **block** | Bu eleman bir blok seviyesinden eleman olarak gösterilecektir, bu eleman önce ve sonra bir satır sonu olacaktır. |
| **inline** | Varsayılı değerdir. Bir satır içi elemandır. |
| **inline-block** | Inline blok elemanı. (CSS2.1 yeni değer) Bir satır içi seviyeden blok konteyner gibi bir öğe görüntüler. Bu bloğun içinde blok düzeyinde kutusu olarak biçimlendirilir ve eleman kendisi bir satır içi seviyeden kutusu olarak biçimlendirilmiş |
| **list-item** | Bu eleman bir liste olarak görüntülenir. |
| **run-in** | Bu eleman bir blok düzeyi elemanları veya satır içi elemanları olarak bağlama göre görüntülenir. |
| **compact** | CSS bir değer kompakt vardır, fakat geniş destek eksikliği, nedeniyle CSS2.1 kaldırılmıştır. |
| **marker** | CSS bir değer işaretleyici vardır, fakat geniş destek eksikliği nedeniyle CSS2.1 kaldırılmıştır. |
| **table** | Bu eleman öncesi ve tablodan sonra bir satır ara ile, (<table> benzer) bir blok düzeyi tablo olarak görüntülenir. |
| **inline-table** | Eleman bir tablo gibi davranır, ancak satır içi dikdörtgen olarak gösterilir. |
| **table-row-group** | Bu eleman görüntülemek için bir veya daha fazla satır olarak gruplandırılmış olacak; Eleman <tbody> elemanı gibi davranır. |
| **table-header-group** | Eleman <thead> elemanı gibi davranır. |
| **table-footer-group** | Eleman <tfoot> elemanı gibi davranır. |
| **table-row** | Bu eleman bir tablo satır olarak gösterilecektir. (benzer <tr>) |
| **table-column-group** | Bu eleman (benzer <COLGROUP>) görüntülemek için bir veya daha fazla sütun bir gruplaşma olarak kullanılacaktır. |
| **table-column** | Bu eleman hücrelerinin bir sütun olarak görüntülenir. (benzer <col>) |
| **table-cell** | Bu eleman bir tablo hücresi olarak görüntülenir. (<td> ve <th> benzer) |
| **table-caption** | Bu eleman bir tablo başlığı olarak görüntülenir.(benzer <caption>) |
| **inherit** | Karşılıklar ana elemandan ekran özelliği değerini miras alır. |

**Display:inline-block Özellik**

**Inline-Block** değeride adı üstünde **inline** ve **block** karışımıdır. Teknik açıdan iki farklı sistemin bir yapı içerisinde kullanılması durumudur. **display:inline-block** tanımı yapılmış elemanlar satır içi akış içinde **satır** içi eleman gibi davranırken hemde **block** olarak Yükseklik ve genişlik değerlerini verebiliyoruz; **fakat tek farkı** vardır. **block** gibi sonrasında boşluk bırakıp alt satıra geçmemektedir.

CSS:

span {

  background-color: #fca64f;

  display: inline-block;

  width: 80px;

  margin:0 10px;

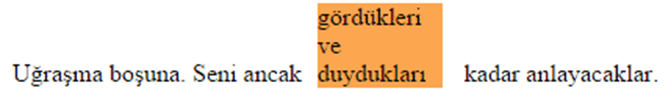
}

HTML:

<p>Uğraşma boşuna. Seni ancak<span> gördükleri ve duydukları</span>

kadar anlayacaklar.</p>

ÇIKTI:



**DİKEYDE HİZALAMA**

Float uygulanmış elemanların dikeyde hizalama çok zordur, aksine **display:inline-block** uyguladığımız elemanları sadece **vertical-align** özelliği kullanarak dikey hizalama yaparız.

**vertical-align** özelliği bir satır içi eleman veya tablo hücresi elemanının içeriğini **dikey hizalama** yapabilir. **vertical-align** satır içi elemanları dikeyde hizaladığı gibi **inline-block** elemanlarında da aynı görevi yapar. **Div** elemanı block elemanıdır; Dikeyde hizalama yapamazsınız fakat **inline-block** elemanı hale dönüştürürseniz dikeyde hizalama yaparsınız.

Yan yana dizilmiş üç kutu hazırlayalım ve bunları kapsayıcı katmana göre dikey olarak üstte, ortada ve altta hizalayalım.

CSS:

.anakutu{

  border: 3px black solid;

  padding: 5px;

  text-align: center;

  width:350px;

}

.solkutu{

  background-color: lightblue;

  display: inline-block;

  height: 100px;

  vertical-align: middle;

  width: 100px;

}

.ortakutu{

  background-color: DarkCyan;

  display: inline-block;

  height: 50px;

  vertical-align: middle;

  width: 100px;

}

.sagkutu{

  background-color: PaleGreen;

  display: inline-block;

  height: 50px;

  vertical-align: bottom;

  width: 100px;

}

HTML:

<div class="anakutu">

    <div class="solkutu">Sol</div>

    <div class="ortakutu">Orta</div>

    <div class="sagkutu">Sağ</div>

</div>

ÇIKTI:



**CSS DİSPLAY:TABLE**

Css tablo modeli **HTML 4** tablo modeline dayalı ve oldukça iyi bir tarayıcı desteğine sahipdir. Her iki tablo modelleri, tablo yapısı ve tablo görüntüsü birebir aynıdır. **CSS**'de tablonun, satırların ve sütunların nasıl oluşturulacağı belirtilmiştir. Tablo konusunu incelediğinize göre **HTML** tablo yapısını bilirsiniz. **CSS** tablo oluşturmasınıda çabuk öğreneceksinizdir. Her **HTML** tablo öğesine eşdeğer **CSS** ekran bir değeri vardır. Tek fark **CSS** tanımlamasında **td** ve **th** arasında ayrım yokdur.

Aşağıda **HTML** Tablo elemanları ve bunlara karşılık gelen **CSS** gösterge değeri vardır.

table    { display: table }

tr       { display: table-row }

thead    { display: table-header-group }

tbody    { display: table-row-group }

tfoot    { display: table-footer-group }

col      { display: table-column }

td, th   { display: table-cell }

caption  { display: table-caption }

Yukarıda ki tanımlamalara bakarak **CSS**'de tablo uluşturmak zor olmaması gerekir. Basit bir örnek verelim.

HTML:

<div id="tablo">

  <div class="tr">

    <span class="td">Hücre 1</span>

    <span class="td">Hücre 2</span>

    <span class="td">Hücre 3</span>

  </div>

  <div class = "tr">

    <span class ="td">Hücre 1</span>

    <span class ="td">Hücre 2</span>

    <span class ="td">Hücre 3</span>

  </div>

  <div class ="tr">

    <span class ="td">Hücre 1</span>

    <span class ="td">Hücre 2</span>

    <span class ="td">Hücre 3</span>

 </div>

CSS:

#tablo {display:table;}

.tr {display:table-row;}

.td {display:table-cell;

     border:1px solid #000;

  }

ÇIKTI:



Yukarıda sadece **HTML** kodlara bakarsanız kolayca **Tekil ve sınıfların** içine **table, tr** ve **td** belirtilmiş **div** ve **span** içinde tekil-sınıf nitelikleri kullandım hariç olarakda Temel Tablo yapısını görebilirsiniz. Dikat ederseniz **CSS** tanımlamasında küçük bir tanımlama ve tanıdık **tablo**, **tablo satırı** ve tablo hücresini **div ve span** etiketlerinde tanımladık.

Tablolarımıza **border, background, width, height** gibi style özellikleri verebiliriz.

CSS tablo sınırları için üç özellik verebiliriz.

·     **border-Collapse :** özelliği, table elementinde satır (tr) ve hücre (td-th) kenarlıklarının bitişik veya ayrı ayrı olmasını ayarlar. İki değer alır.

o  **separate :** Satır ve hücreler, birbirlerinden **ayrı** kenarlığa sahiptir.

o  **collapse :** Satır ve hücreler, birbirleri ile bitişik kenarlıkları **ortak** kullanırlar.

·       **borderSpacing :** özelliği, **table** ve **inline-table** elementlerinde yatay ve dikey hücre kenarlık boşluklarının belirlenmesini sağlar. Bu özellik sadece **border-collapse** özelliğinin değeri **"separate"** olarak belirtildiği zaman kullanılabilir.

·     **empty-cells :** Boş hücrelerin görünüp görünmemesini sağlar. Bu özellik sadece **border-collapse** özelliğinin değeri **"separate"** olarak belirtildiği zaman kullanılabilir. **show** değeri verildiğinde hücreyi gösterir ve **hide** değeri verildiğinde gizler.

iki sütunlu bir tablo örneği:

CSS:

#table {

  display:table;

}

.tr {display:table-row;}

.td {

    display:table-cell;

    width:250px;

    border:1px solid #7f8283;

    text-align: justify;

    background-color:#d2f0f8;

}

HTML:

<div id="table">

<div class="tr">

  <span class="td">

Lorem Ipsum is simply dummy text of the printing and typesetting industry.

Lorem Ipsum has been the industry's standard dummy text ever since the 1500s,

when an unknown printer took a galley

</span>

 <span class="td">

Lorem Ipsum is simply dummy text of the printing and typesetting industry.

Lorem Ipsum has been the industry's standard dummy text ever since the 1500s,

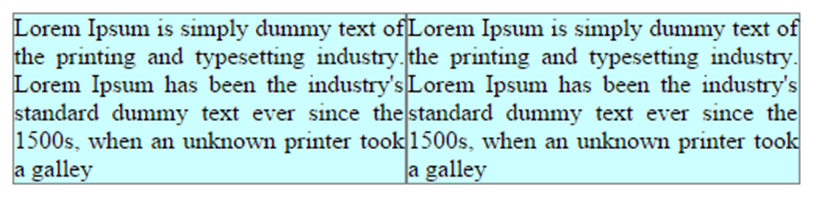
when an unknown printer took a galley

 </span>

<div class="tr">

</div>

ÇIKTI:



Yukarıdaki örnekte görüldüğü gibi hücreler birleşik hücreleri ayırmak için **margin** kullanamayız; çünkü **table-cell** "td" gibi davranmaktadır. Bunun için tablo elemanlarında kullandığımız **border-spacing** komutunu kullanmak. **border-spacing**’i sadece display: table elemanına verebiliyoruz. yukarıda ki örneği yeni duruma göre ayarlayalım.

CSS:

#table {

  display:table;

  border-spacing: 10px;

}

.tr {display:table-row;}

.td {

    display:table-cell;

    width:250px;

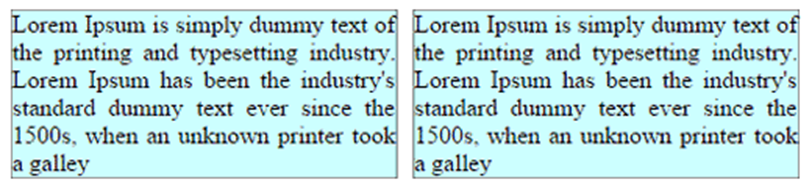
    border:1px solid #7f8283;

    text-align: justify;

    background-color:#d2f0f8;

}

ÇIKTI:



Üç sutunlu Tablo yapalım ve 200x140 uçak resminide yerleştirelim:

CSS:

#table {

  display:table;

  border-spacing: 10px;

}

.tr {display:table-row;}

.td {

    display:table-cell;

    width:250px;

    border:1px solid #7f8283;

    text-align: justify;

    background-color:#d2f0f8;

}

img{ margin:10% 10% ;}

HTML:

<div id="table">

  <div class="tr">

<span class="td">

  <img src="ucak.jpg" alt=""/>

<p>Lorem Ipsum is simply dummy text of the printing and typesetting industry.

Lorem Ipsum has been the industry's standard dummy text ever since the 1500s,

when an unknown printer took a galley</p>

</span>

<span class="td">

  <img src="ucak.jpg" alt=""/>

<p>Lorem Ipsum is simply dummy text of the printing and typesetting industry.

Lorem Ipsum has been the industry's standard dummy text ever since the 1500s,

when an unknown printer took a galley</p>

</span>

<span class="td">

<img src="ucak.jpg" alt=""/>

<p>Lorem Ipsum is simply dummy text of the printing and typesetting industry.

Lorem Ipsum has been the industry's standard dummy text ever since the 1500s,

when an unknown printer took a galley</p>

</span>

  <div class="tr">

</div>

ÇIKTI:



**CSS display:none özelliği**

CSS'de belkide en çok kullanılan özelliklerden biride **display: none** özelliğidir. **display: none** bir elemanı gizler ve sayfada yer teşkil etmez diğer elemanlar gizlenen eleman yokmuş gibi sayfada yer teşkil eder. Genelde **javascript** ile birlikte kullanılır. Animasyonlarda, Tab türü menülerde makale yazılmış div'leri gizlemede kullanılır.

CSS:

.gizle{display:none}

HTML:

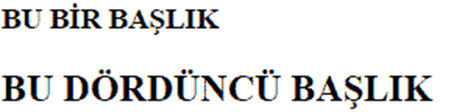
<h3>BU BİR BAŞLIK</h3>

<h3 class="gizle">BU İKİNCİ BİR BAŞLIK</h3>

<h3 class="gizle">BU ÜÇÜNCÜ BİR BAŞLIK</h3>

<h2>BU DÖRDÜNCÜ BAŞLIK</h2>

ÇIKTI:



**PSEUDO SINIFLAR**

Pseudo sınıf ve elementleri CSS’i destekleyen web tarayıcıları tarafından otomatik olarak tanınan özel sınıf ve elementlerdir. Bu sınıf ve elementler (x)html hiyerarşisi ile erişemediğimiz element ve sınıflara erişmemizi sağlar. **Pseudo sınıfı** bir elementi farklı sınıflara böler(örn: link elementini active, visited vd. sınıflarına böler) **Pseudo elementi** ise bir elementi alt kısımlara böler (örneğin bir paragrafın ilk harfi, bir paragrafın ilk satırı gibi.) Pseudo sınıfına örnek:

a**:visited** {

**color**: red;

}

Pseudo elementine örnek:

p**:first-line** {

**font-weight**: bold;

}

Pseudo sınıf ve elemtleri HTML class özelliği olarak belirtilmemiştir. Normal sınıflar pseudo sınıf ve elementleri ile kullanılabilir.

a**.disariyalinkller:link,**

a**.disariyalinkller:visited** {

**color**: maroon;

}

Aynı şekilde id seçicileri ile birlikte de kullanılabilirler

a**#altkisim:link** {

**font-weight**: bold;

}

Pseudo sınıflarını da ikiye ayıra biliriz. **Link Pseduo Sınıfıları** ve **Dinamik Pseudo Sınıfları**

**Link Pseudo sınıfı**

Yanlızca linklere uygulanan iki Link Pseduo sınıfı vardır. **:link** ve **:visited** **:link -**Ziyaret edilmemiş sayfanın linkine stil tanımlaması yapmak için kullanılır. Ancak bir çok web tarayıcısı yapılan stil tanımlarını ziyaret edilmiş sayfa linklerine de uygular. **:visited -** Henüz ziyaret edilmiş sayfa linklerine stil tanımlaması yapmak için kullanılır.

a**:link** {

**color**: blue;

}

a**:visited** {

**color**: red;

}

Bunun yerine genelde aşağıdaki gibi bir kod da kullanılır

a {

**color**: blue;

}

a**:visited** {

**color**: red;

}

Bu kodlama ile kullanıcaya ziyaret ettiği sayfa linkleri farklı renkte gösterilerek içeriksel bir bilgi görsel olarak verilebilir.

**Dinamik Pseudo Sınıfları**

**Dinamik Pseudo Sınıfları** sayfa görünümüne çok büyük katkılar yapabilir. Bu sınıflar genelde linklere uygulanır ancak bir çok kullanım alanları vardır. tanedir **:focus**, **:active** ve **:hover**

**Not** Pseudo sınıflarında sıralama önemlidir. Genel kullanımda **“link-visited-hover-active,”**sıralaması yapılmalıdır. Tüm sınıflar içinse **“link-visited- focus-hover-active.”** sıralaması yapılmalıdır.

**:focus -** Odaklanan elemente stil tanımlası yapmak için kullanılır Örn: input alanına içerik girerken **:active** - Aktif olan elemente stil atamak için kullanılır. **:hover -** Bir elementin üzerine Farenin imleci geldiğinde yapılacak tanımlama için Örn: bir linkin üzerine fare ile geldiğimizde renk değiştirmesini sağlamak için Linkler için genel kullanım

a**:link** {

**color**: navy;

}

a**:visited** {

**color**: gray;

}

a**:hover** {

**color**: red;

}

a**:active** {

**color**: yellow;

}

**:focus** için bir örnek verecek olursak

input**:focus** {

**background**: silver;

**font-weight**: bold;

}

**İlk Çocuk Elementi seçmek**

Diğer bir pseudo sınıfımız **:first-child**‘dır. **first:child:** Belirtilen elementin ilk Çocuk Elementine stil atamak için kullanılır.

p:first-child {

    font-weight: bold;

}

li:first-child {

    color:#f00;

}

HTML;

<div>

    <p>Bu paragraf ilk çocuk elementidir ve sonuç olarak kalın olacaktır</p>

    <ul>

        <li>Bu liste ilk çocuk elementidir ve font rengi kırmızı olacak</li>

        <li>Bu <strong>çocuk element </strong>değil</li>

        <li>Bu da değil</li>

    </ul>

    <p>Bu pragraf <em>bir</em> çocuk element değil.</p>

</div>

**Pseudo Element Seçicileri**

Bu elementler hayali elementlerdir. (X)HTML hiyerarşisi içinde bu elemntlere yoktur. CSS2.1 de Dört adet Pseudo Element Seçicisi vardır: **first-letter**, **first-line**, **before** ve **after**

**first-letter (ilk harf)**

Bir blok-level elementin ilk harfine stil tanımlması yapmak için kullanılır. Örnek verecek olursak h1 elementinin baş harfinin büyük olması için :

h1**:first-letter** {

**font-size**:200%;

}

<h1>Bu büyük bir başlık</h1>

**first-line (ilk satır)**

Bir metnin ilk elementine stil atamak için kullanılır. Örneğin paragraflarınızın ilk satırlarını renklendirmek isitiyorsunuz.

p**:first-line** {

**color**: red;

}

**before ve after elementleri**

Bir elementin öncesine ve sonrasına bir içerik veya özellik eklemek için kullanılır. Notlarımızın başına otomatik oalrak Not yazmak için

p**.not:before** {

**content**: "Not."

}

kodu yeterlidir. Aynı şekilde paragraflarımzın sonuna otomatik bitti yazıp nokta koymak istersek

body**:after** {

**content**: " Bitti.";

}

**FLEXBOX**

Flex özellikle karmaşık projelerde web geliştiricilerinin işini kolaylaştıran, web sitesindeki elamanları yatayda ve dikeyde hizalamaya yarayan ve günümüzde oldukça popüler olan CSS layoutlarından biridir.

**Neden Flexbox?**

·       Flex ile esnek bir yapı kurarak tasarladığınız siteyi responsive yapmanız çok daha kolaydır.

·       Dikeyde ortalama flex yapısıyla çok kolaydır.

·       Elemanların genişliğini ve yüksekliğini kolayca boyutlandırabilirsiniz.

·       Günümüzde tüm güncel tarayıcılar tarafından desteklenmektedir.

**Flexbox Nasıl Kullanılır?**

Flexbox’a başlamak için önce flex konteynere ihtiyacımız vardır.

.flex-container {

  display: flex;

}

Bu makalede bahsedeceğimiz flex özellikleri:

·       justify-content

·       align-items

·       flex-direction

·       flex-wrap

·       flex-flow

·       align-content

·       order

**justify-content Özelliği**

justify-content Özelliği içindeki elemanların X ekseninde yani yatayda hizalamaya yarar. Şu değerleri alabilir:

·       **flex-start**: Öğeleri flexbox container’ın sol tarafına hizalar.

·       **flex-end**: Öğeleri flexbox container’ın sağ tarafına hizalar.

·       **center**: Öğeleri flexbox container’ın ortasına hizalar.

·       **space-between**: Öğeler eşit aralıklarla yatay şekilde yayılır

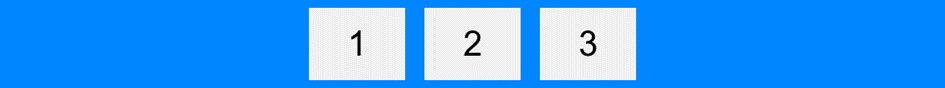
·       **space-around**: Öğeler etraflarında eşit aralıklar olacak şekilde yatay şekilde yayılır

.flex-container {

  display: flex;

  justify-content: center;

}



*justify-content: center özelliği öğeleri yatayda ortalar.*

**align-items Özelliği**

align-items Özelliği içindeki elemanların Y ekseninde yani dikeyde hizalamaya yarar. Şu değerleri alabilir:

·       **flex-start**: Öğeleri flexbox container’ın tepesine hizalar

·       **flex-end**: Öğeleri flexbox container’ın aşağısına hizalar

·       **center**: Öğeleri flexbox container’ın dikey ortasına hizalar

·       **baseline**: Öğeleri flexbox container’ın yazı referans çizgisine(baseline) hizalar

·       **stretch**: Öğeler flexbox container boyunca uzarlar

.flex-container {

  display: flex;

  align-items: center;

}



*align-items: center özelliği öğeleri dikeyde ortalar.*

**flex-direction Özelliği**

flex-direction özelliği öğelerin hangi yönde yerleştireleceğini belirler. Şu değerleri alabilir:

·       **row**: Öğeler yazı yönü ile aynı yönde yerleştirilir.

·       **row-reverse**: Öğeler yazı yönünün tersi yönünde yerleştirilir.

·       **column**: Öğeler yukarıdan aşağıya doğru yerleştirilir.

·       **column-reverse**: Öğeler aşağıdan yukarıya doğru yerleştirilir.

.flex-container {

  display: flex;

  flex-direction: column;

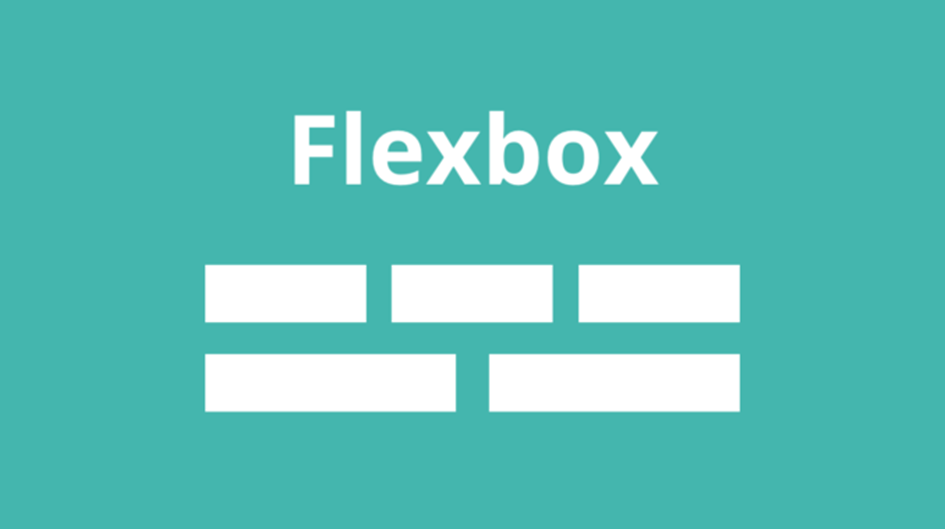
}



*flex-direction: column özelliği öğeleri yukarıdan aşağıya sıralamaya yarar.*

**flex-wrap Özelliği**

flex-wrap özelliği birden fazla satırda gösterilecek öğelerin nasıl bir davranış sergileyeceğini belirlemek için kullanılır. Şu değerleri alabilir:



·       **nowrap**: Tüm öğeler tek satıra sığar.

·       **wrap**: Öğeler satırlara bölünür.

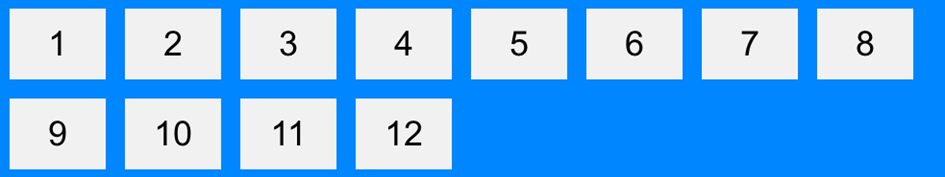
·       **wrap-reverse**: Öğeler satırlara ters sırada bölünür.

.flex-container {

  display: flex;

  flex-wrap: wrap;

}



*flex-wrap: wrap özelliği öğeleri birden fazla satıra böler.*

**flex-flow Özelliği**

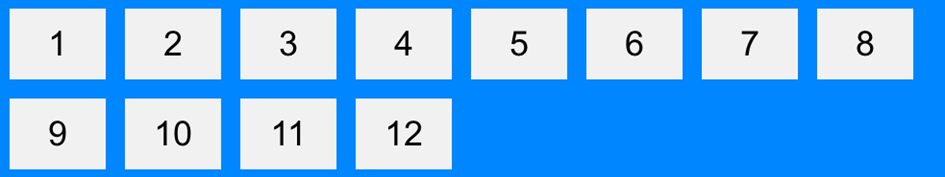
flex-direction ve flex-wrap kuralları, flex-flow kısa yazım kuralı ile birlikte yazılabilirler. Bu kısa yazım kuralı, iki kuralın değerlerini aralarında boşluk ile birlikte alır. Bu kısa yazım kuralı, iki kuralın değerlerini aralarında boşluk ile birlikte alır.

.flex-container {

  display: flex;

  flex-flow: row wrap;

}



*flex-flow: row wrap özelliği öğeleri hem satır düzeninde hem de sığmayan elemanları bir alt satıra atmayı sağlar.*

**align-content Özelliği**

align-content kuralı ile satırları birbirinden ayırabilirsiniz. Bu kural şu değerleri alır:

·       **flex-start**: Satırlar flex container’ın tepesinde toplanır.

·       **flex-end**: Satırlar flex container’ın aşağısında toplanır.

·       **center**: Satırlar flex container’ın dikey ortasında toplanır.

·       **space-between**: Satırlar aralarında eşit boşluk kalacak şekilde yayılırlar.

·       **space-around**: Satırlar etraflarında eşit boşluk kalacak şekilde yayılırlar.

·       **stretch**: Satırlar flex container’ı kaplamak için uzarlar.

.flex-container {

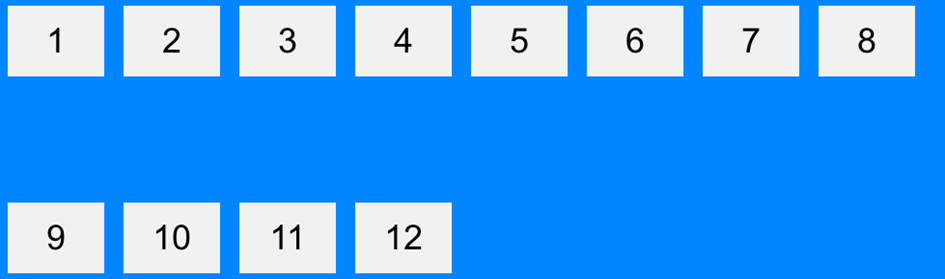
  display: flex;

  flex-wrap: wrap;

  align-content: space-between;

}

Bu kafa karıştırıcı olabilir, ama align-content satırlar için aralıkları belirlerken, align-items öğelerin flex container içerisinde beraberce nasıl hizalanacağını belirler. Eğer tek satır varsa, align-content kuralının bir etkisi yoktur.



*align-content: space-between özelliği satırların arasında eşit boşluk kalacak şekilde genişletir.*

**order Özelliği**

Bazen row ya da column yerleşimini tersine çevirmek yeterli olmaz. Bu gibi durumlarda order kuralı ile öğeleri tek tek sıralandırabiliriz. Öğelerin başlangıç sıra değeri 0’dır, ama biz bu değeri pozitif ya da negatif bir değere eşitleyebiliriz.

<div class="flex-container">

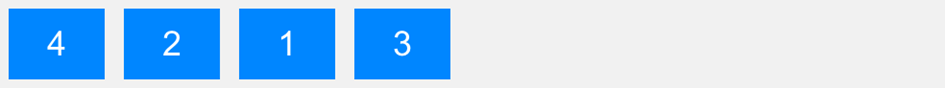
  <div style="order: 3">1</div>

  <div style="order: 2">2</div>

  <div style="order: 4">3</div>

  <div style="order: 1">4</div>

</div>



*order özelliği öğeleri manuel sıralamaya yarar.*

**CSS Borders**

Border elementlerimize verdiğimiz kenarlıktır. Çerveve çizgisi gibi düşünebiliriz. Border 3 adet özellik almaktadır.Bunlar  **border-style, border-width ve border-color özellikleridir.**

**border-style**

Çizgimizin türünü temsil etmektedir.Alacağı değerker aşağıdaki gibidir

·       dotted – Noktalı çizgi

·       dashed – Kesik çizgilerden oluşan çizgi

·       solid – Düz çizgi

·       double – Çift çizgi

·       groove – 3 boyutlu oluklu çizgi. Bu efekt border rengine göre ortaya çıkar.

·       ridge – Çıkıntılı çizgi. Bu efekt border rengine göre ortaya çıkar.

·       inset – İçe doğru 3 boyutlu görünen çizgi. Bu efekt border rengine göre ortaya çıkar.

·       outset – Dışa doğru 3 boyutlu çizgi. Bu efekt border rengine göre ortaya çıkar.

·       none – Kenar çizgisi yok

·       hidden- Gizli kenar çizgisi

**Örnek**

<body>

    <p class="solid">Border nedir nasıl kullanılır.</p>

    <p class="dotted">cetinyazilim.com</p>

    <p class="dashed">Yazılım Mühendisliği</p>

    <p class="groove">Html-Css-Bootstrap-Javascript</p>

</body>

<style>

        .solid {

            border-style: solid;

        }

        .dotted {

            border-style: dotted;

        }

        .dashed {

            border-style: dashed;

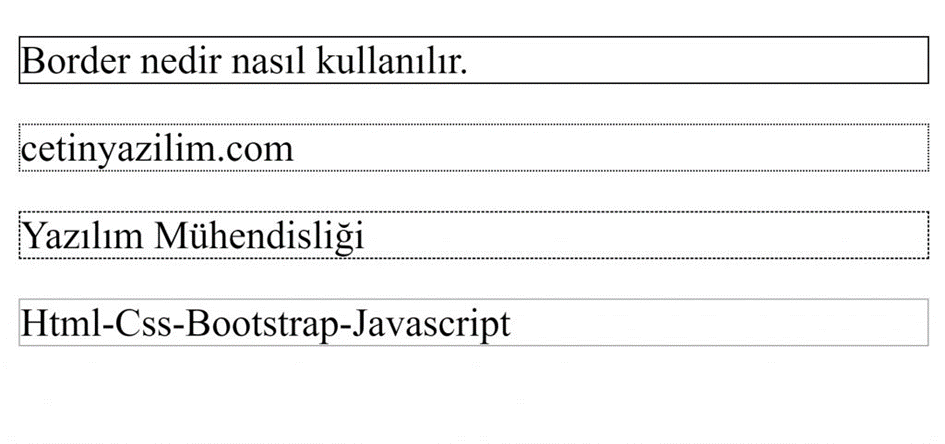
        }

        .groove {

            border-style: groove;

        }

    </style>



**border-width**

Kenarlığımızın kalınlığı belirler.Varsayılan olarak sıfırdır.

<body>

    <p class="solid">Border nedir nasıl kullanılır.</p>

    <p class="dotted">cetinyazilim.com</p>

    <p class="dashed">Yazılım Mühendisliği</p>

    <p class="groove">Html-Css-Bootstrap-Javascript</p>

</body>

<style>

        .solid {

            border-style: solid;

            border-width: 3px;

        }

        .dotted {

            border-style: dotted;

            border-width: 6px;

        }

        .dashed {

            border-style: dashed;

            border-width: 9px;

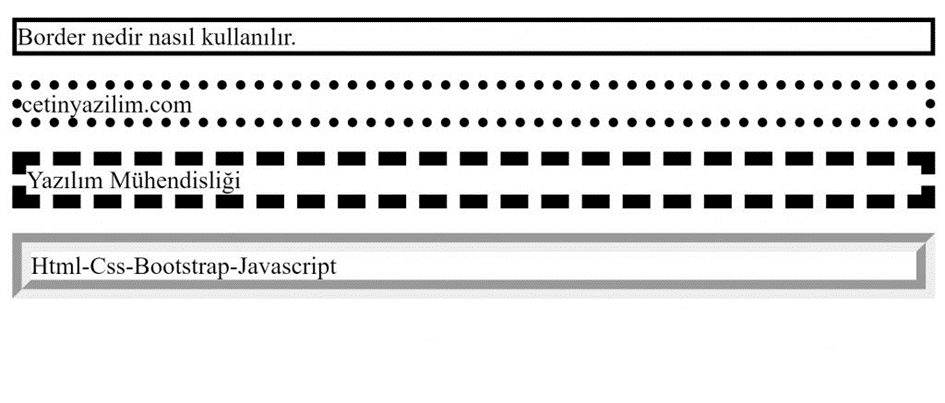
        }

        .groove {

            border-style: groove;

            border-width: 12px;

        }



**border-color**

Kenarlığımızın rengini belirler.

<body>

    <p class="solid">Border nedir nasıl kullanılır.</p>

    <p class="dotted">cetinyazilim.com</p>

    <p class="dashed">Yazılım Mühendisliği</p>

    <p class="groove">Html-Css-Bootstrap-Javascript</p>

</body>

<style>

        .solid {

            border-style: solid;

            border-width: 3px;

            border-color:tomato;

        }

        .dotted {

            border-style: dotted;

            border-width: 6px;

            border-color:green;

        }

        .dashed {

            border-style: dashed;

            border-width: 9px;

            border-color:yellow;

        }

        .groove {

            border-style: groove;

            border-width: 12px;

            border-color:purple;

        }

    </style>



**Örnek**

Aşağıdaki örnekte farklı renkler yazaraktan renklendirme yapabiliriz.

<body>

    <p class="solid">Border nedir nasıl kullanılır.</p>

</body>

<style>

        .solid {

            border-style: solid;

            border-width: 5px;

            border-color: red green blue yellow;

        }

    </style>



**Önemli**

Şimdiye kadar gördüğümüz bu 3 özelliği tek satırda vermemiz mümkündür.

<body>

    <p>Border nedir nasıl kullanılır.</p>

</body

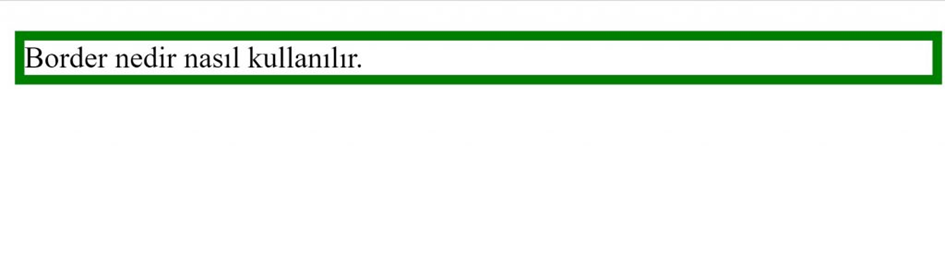
<style>

        p {

            border: 5px solid green;

        }

    </style>

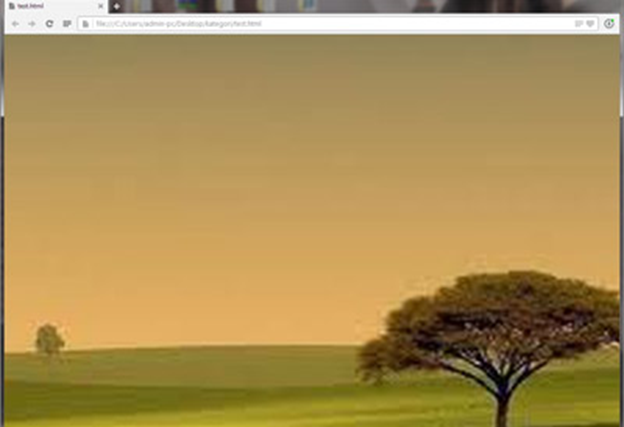


**ARKA PLAN RESMİ**

**background-image** özelliği ve ögelerine istediğimiz zemin resmi eklemek ve konumlandırmak için kullanılırız. Örnek olarak "zemin.jpg" isimli görselimizi zemin resmi olarak sayfamıza eklemek için aşağıdaki stil şablonu kodunu uygulamalıyız.

  body {background-image:url(zemin.jpg);}

ÇIKTI:



**ZEMİN RESMİ TEKRARI**

**background-repeat** özelliği zemine eklenen görselin tekrarlanma biçimini kontrol etmek için kullanılır. Dört ayrı değer alır:

·       **background-repeat:repeat;** (Görselin hem yatay olarak hem de dikey olarak tekrar etmesini istiyorsak bu özelliği kullanırız. Bu özellik yazılmasa bile CSS varsayılan olarak bunu yapacaktır.)

·       **background-repeat:no-repeat;**(Eklediğimiz görselin hiçbir şekilde tekrar etmesini istemiyorsak bu özelliği kullanırız.)

·       **background-repeat:repeat-x;** (Eklediğimiz görsel yatay olarak yani x ekseni boyunca tekrar etsin istiyorsak bu özelliği kullanırız.)

·       **background-repeat:repeat-y;** (Eklediğimiz görsel dikey olarak yani y ekseni boyunca tekrar etsin istiyorsak bu özelliği kullanırız.)

ÖRNEK 1:

body{background-image:url(logo.png);

     background-repeat : repeat;

     }

ÇIKTI:



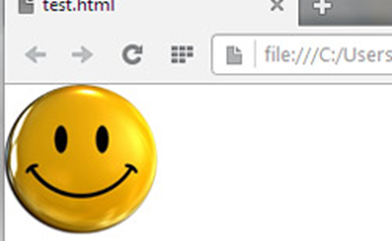
ÖRNEK 2:

body{background-image:url(logo.png);

     background-repeat :no-repeat;

     }

ÇIKTI:



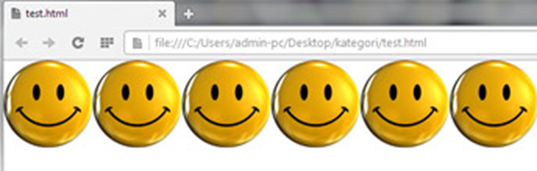
ÖRNEK 3:

body{background-image:url(logo.png);

     background-repeat :repeat-x;

     }

ÇIKTI:



ÖRNEK 4:

body{background-image:url(logo.png);

     background-repeat :repeat-y;

     }

ÇIKTI:



**ZEMİN RESMİ POZİSYONU**

**"background-position"** özelliği görselin sayfadaki pozisyonunu belirler. Dikey hizalamada **üst (top)**, **orta (center)** ve **alt (bottom)** geçerli bir pozisyonken; yatay hizalamada **sol (left)**, **orta (center)** ve **sağ (right)** geçerlidir. Bu özellik için 11 ayrı değer bulunmaktadır.

·       **background-position:top left;**(Bu bildirimi kullandığımızda görselimiz üstün solunda yer alacak.)

·       **background-position:top center;**(Bu bildirimi kullandığımızda görselimiz üstün ortasında yer alacak.)

·       **background-position:top right;**(Bu bildirimi kullandığımızda görselimiz üstün sağında yer alacak.)

·       **background-position:center left;**(Bu bildirimi kullandığımızda görselimiz ortanın solunda yer alacak.)

·       **background-position:center center;**(Bu bildirimi kullandığımızda görselimiz ortanın ortasında yer alacak.)

·       **background-position:center right;**(Bu bildirimi kullandığımızda görselimiz ortanın sağında yer alacak.)

·       **background-position:bottom left;**(Bu bildirimi kullandığımızda görselimiz altın solunda yer alacak.)

·       **background-position:bottom center;**(Bu bildirimi kullandığımızda görselimiz altın ortasında yer alacak.)

·       **background-position:bottom right;**(Bu bildirimi kullandığımızda görselimiz altın sağında yer alacak.)

·       **background-position:x% y%;**(Görselimize yatay ve dikey olarak yüzde üzerinden pozisyon vermemizi sağlar. ilk yüzde yatayın, ikinci yüzde ise dikeyin pozisyonudur. Örnek:**background-position:80% 55%;** )

·       **background-position:xpos ypos;**(Görselimize yatay ve dikey olarak pixel bakımından pozisyon vermemizi sağlar. İlk pixel'li rakam yatayın, ikinci pixel'li rakam ise dikeyin pozisyonudur. Örnek:**background-position:150px 200px;** )

Örnek olarak bir ağaç resmini merkez sağa yerleştirelim:

body {width:100%; height:650px;

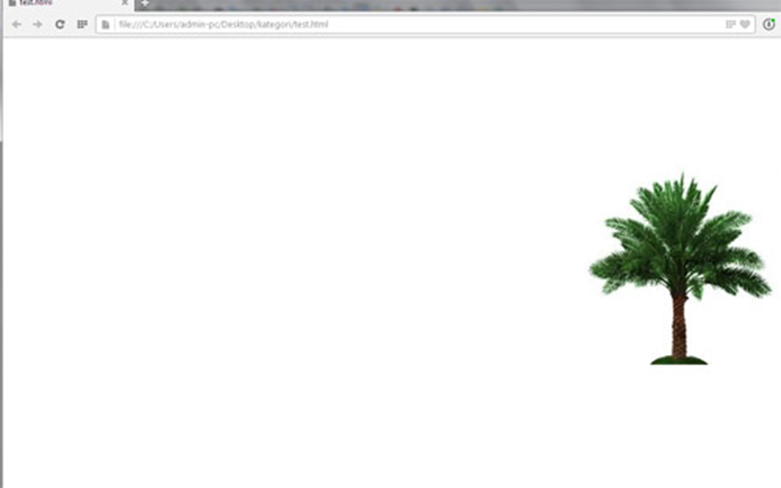
      background-image: url(agac.jpg);

      background-repeat : no-repeat;

      background-position:center right;

      }

ÇIKTI:



Şimdide ikinci örnek olarak ağaç resmini alt merkeze yerleştirelim:

body {width:100%; height:650px;

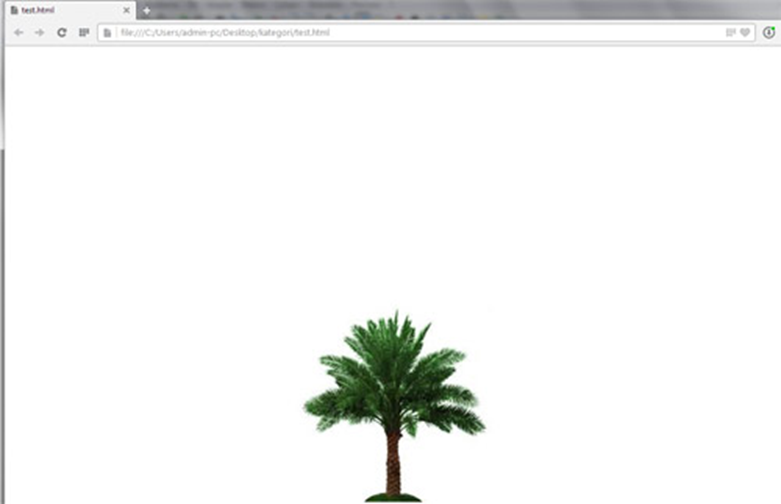
      background-image: url(agac.jpg);

      background-repeat : no-repeat;

      background-position:bottom center;

       }

ÇIKTI:



**"background" özelliklerini kısaltma uygulayarak şöyle de yazabilirdik:**

body{ background:url(logo.gif) no-repeat top center;}

Birkaç örnek daha yapalım:

body{ background: url(logo.gif) no-repeat bottom right;}

 body{ background: url(logo.gif) no-repeat 60% 75%;}

 body{ background: url(logo.gif) no-repeat 100px 200px;}

**background-attachment özelliği**

Zemine eklenen resim sayfa ile beraber kaymasını veya sayfanın zeminine sabit kalmasını sağlar.

1.    scroll: Sayfa hareket ettirildiğinde zeminin de kaymasını sağlar.

2.    fixed: Sayfa hareket ettirildiğinde zeminin sabit kalmasını sağlar.

Hiçbir Şey yazılmadığında default(Geçerli) değer; scroll'dur

body {background-image: url(logo.gif);

      background-repeat : no-repeat;

      background-attachment: fixed }

**CSS Position**

Css position özelliği nedir ve nasıl kullanılır ? Css' de relative, absolute, fixed ve sticky konumlandırma nasıl kullanılır?

Css' de **nesneleri konumlandırmak** için kullandığımız yöntemlerden biri **floating,**bir diğeri ise **position**özelliğidir.

Tüm html etiketleri başlangıç ayarı olarak belli gruplara aittir; **yani Inline ya da Block etiket**grubundalardır. Dolayısıyla bu grupların davranış özelliklerine göre nesneler birbirlerinden etkilenerek sayfa üzerinde görünürler. Yani yan yana gelmeleri ya da alt alta gelmeleri gibi.

Peki bu durum bize yeterli gelmez ve bazı nesnelerin konumlarını değiştirmek isteriz. Bu durumda bir önceki dersimizde floating ile nesneleri yan yana getirebildik.

Floating özelliğinde gördüğümüz gibi nesneler hep birbiri ardına gelerek kendilerine bir konum oluştururlar. Örneğin nesnelere **float:left;** dediğimizde sol tarafa doğru birbiri ardına gelirler.

\*\* Nesnelerin bu şekilde birbirinden etkilenerek sayfa üzerinde konumlanmasına html etiketlerinin **normal akış davranışı** denir.

Biz her zaman normal akış içinde davranmak istemeyiz. Örneğin bir nesneyi sayfa üzerindeki diğer nesnelerden bağımsız olarak başka bir konuma getirmek isteyebiliriz ya da bir <div> etiketini sayfanın bir konumunda sabitleyerek durmasını isteyebiliriz. Bu gibi durumlarda **css position** özelliğini kullanmalıyız.

Css Position' ın alabileceği farklı değerler vardır.

Position : **static | relative | fixed | absolute | sticky**

**Position Static Nedir ve Nasıl Kullanılır ?**

Html etiketleri başlangıç ayarı yani varsayılan olarak static biçimde davranış gösterirler. Yani normal akış içinde birbirlerinden etkilenerek davranış gösterirler. \*\* Static nesneler için **top, right, bottom ve left** konumlandırma özellikleri kullanılamaz.

**Position Relative Nedir ve Nasıl Kullanılır ?**

Bir nesneye **position:relative;** dediğimizde nesne gene **normal akış içerisinde** olmaya devam eder ancak **top, right, bottom ve left**özelliklerini kullanma fırsatı verir.

<div id="first" **class**="box"></**div**>

<div id="second" **class**="box"></**div**>

<div id="third" **class**="box"></**div**>

.box{

**width**: 150px;

**height**: 150px;

**margin**: 5px;

}

#first{

**background**: red;

}

#second{

**background**: blue;

**position**:relative;

**top**:50px;

}

#third{

**background**: green;

}

**#second div'**ine verdiğimiz **position:relative;** özelliğinden dolayı **top:50px;** kullanımını yapabildik ve nesne yukarıdan yani **olduğu konumdan itibaren** **50px aşağıya iner.** Ancak sonrasında gelen elemanları **aşağıya itmediğine ve mevcut konumundan itibaren 50px aşağıya indiğine** dikkat ediniz.

ve aynı şekilde nesneyi 4 köşeden konumlandırabiliriz.

Burada z-index özelliğini de kullandık. **z-index özelliği css** 'de nesnelerin hangisinin önde hangisinin arkada olduğuyla alakalıdır. Yani örneğimizde varsayılan olarak mavi kutu yeşil kutunun üstünde görünürken mavi kutu için **z-index:-1;**dediğimizden dolayı yeşil kutunun arkasına gider.

\*\* Yani z-index sayısal olarak değer alır ve sayısal büyüklüğe göre ya arkaya ya da öne gelir.

**Position Absolute Nedir ve Nasıl Kullanılır ?**

Bir nesnenin position özelliğini absolute yaptığımızda o nesne **normal akış içinden çıkar** ayrıca **4 köşeden konumlandırma** yapmamızı sağlar. Relative ile Absolute nesnelerin farkı **normal akış içinde olup olmadığı** farkıdır.

#second{

**background**: blue;

**position**:absolute;

**top**:50px;

**left**:50px;

}

Örnekte görüldüğü gibi mavi kutu normal akışın dışına çıktığından dolayı diğer iki kutu mavi kutuyu görmezde gelir ve mavi kutunun olduğu yeri doldururlar.

Mavi kutu normal akış içinde olmadığından dolayı konumlandırma başlangıcını sayfanın**sol üst**köşesinden yapar. Aslında burada sol üst köşe derken body' den bahsediliyor. Çünkü her hangi bir absolute nesne kendini konumlandırmak için bir relative nesnenin kapsamında olmayı bekler ki burada kapsayıcı hiç bir eleman olmadığından dolayı body mavi kutu için bir kapsayıcı elemandır ve konumlandırmasını body 'e göre yapar.

Örneğimizi şu şekilde değiştirdiğimizde artık konumlandırmamızı body' e göre değil de başka bir kapsayıcı elemana göre yapmış oluruz ancak bu kapsayıcı elemanın relative bir nesne olma zorunluğu vardır.

<**div** id="first" class="box"></**div**>

<**div** id="second" class="box">

  <**div** id="inner" class="inner-box">

  </**div**>

</**div**>

<**div** id="third" class="box"></**div**>

.box{

**width**: 150px;

**height**: 150px;

**margin**: 5px;

}

#first{

**background**: red;

}

#second{

**background**: blue;

**position**:relative;

**top**:50px;

**left**:50px;

}

#third{

**background**: green;

}

.inner-box{

**width**: 50px;

**height**: 50px;

**background-color**:white;

}

#inner{

**position**:absolute;

**top**:10px;

**left**:20px;

}

Burada #inner ismindeki div absolute olarak belirtildi ancak dikkat edersiniz ki bu div 'in bir kapsayıcı elamanı var ve bu kapsayıcı eleman relative bir nesne olduğundan dolayı konumlandırmasını bu mavi kutunun sol üst köşesinden başlayarak yapar aksi halde body etiketinden itibaren konumlandırılma yapılırdı.

\*\* Mavi kutuyu sayfada nereye götürürsek götürelim beyaz kutu mavi kutu içinde hep aynı konumda kalmaya devam eder.

**Position Fixed Nedir ve Nasıl Kullanılır ?**

**Css  fixed**' in ne olduğunu anlamak için ilk önce html' de sıklıkla duyduğumuz **viewport kavramın ne olduğunu** ve hangi amaçla kullanıldığını bilmeliyiz.

**Viewport:**Sayfanın o anki görüntü alanına viewport denir. Yani 1200px' lik toplam yüksekliğe ve 800px genişliğe sahip olan bir web sitesinin küçük bir tarayıcıda açıldığını düşünelim ve bu tarayıcı bize sitenin sadece 600px lik yükseklik ile 400px' lik genişliğini gösterebiliyorsa o tarayıcının viewport' u 600 x 400 px diyebiliriz.

Viewport' un ne olduğunu anladıktan sonra position fixed 'in ne olduğunu daha iyi anlayabiliriz. Tarayıcının viewport' una bağlı olarak sabit bir yerde konumlanan nesneler oluşturmak istiyorsak position fixed kullanmamız gerekiyor. Tarayıcının viewport' una bağlı olarak derken kastedilen yatay ve dikey kaydırma çubuklarının kullanılmasına rağmen nesnenin konumu hep sabit kalmasındandır. Bizim için başlangıç noktası o anki viewport' dur.

<div **class**="fixed">

  This element has position: fixed;

</**div**>

.fixed {

**position**: fixed;

**bottom**: 0;

**right**: 0;

**width**: 250px;

**border**: 4px solid red;

}

Örneğimizde **.fixed sınıfı**na sahip olan eleman **sayfanın hep sağ alt köşesinde sabit kalır.** Kaydırma çubuklarını kullanmış olduğumuzda oluşan her viewport alanına göre tekrar konumlandırma yapılır ki; bu konumlandırma zaten sayfanın sağ alt köşesidir.

**Position Sticky Nedir ve Nasıl Kullanılır ?**

Position: sticky kaydırma çubuklarını takip eder ve istenilen pozisyon yakalandığında nesneyi o noktada sabitler. Örneğin bir nesne için yukarıdan 0 konumuna geldiğimiz anda nesnenin sabitlenmesini isteyebiliriz. Genelde menüler için sticky postion kullanılır.

<div **class**="sticky">

  This element has position: sticky;

</**div**>

.sticky {

**position**: sticky;

**top**: 0;

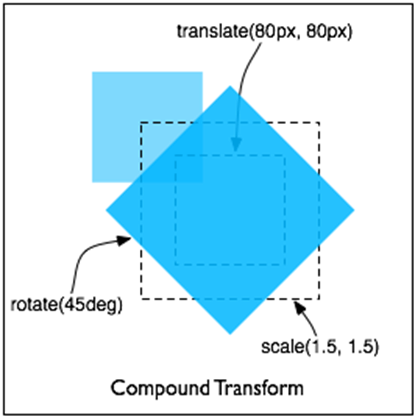
**width**: 250px;

**border**: 4px solid red;

}

**TRANSFORM**

Webkit ekibinin ilk olarak 2008 yılında CSS’e kattığı bu özellik daha sonra w3c tarafından standartlara girdi. HTML elemanlarını döndürmek, boyutlandırmak, eğmek gibi eylemleri gerçekleştirmemize yarıyor. 2d Transform özelliği 3d’ye göre tarayıcı desteği konusunda çok avantajlı. 2B Dönüştürme özelliklerini ie9+ desteklerken, 3B Dönüştürme özelliklerini henüz betası çıkmış olan ie10+ destekliyor.



Bu makalede CSS3 transform 2 Boyutlu Dönüştürme konusunu inceleyeceğiz. 3 Boyutlu Dönüştürme konusuna daha sonra değineceğim.

**2 Boyutlu Dönüştürme(2d Transform)**

CSS 2d Transform özelliği  iki boyutlu düzende elemanları dönüştürmeye olanak sağlıyor. Tabi böyle tanımlama yaparak anlamak zor en kolayı özellikleri tanımlayıp örnek yapmak.

Dönüştürme tanımları 3 bölümde inceleniyor.

* transform Özelliği
* transform-origin özelliği
* Dönüştürme fonksiyonları (transform özelliği değerleri – translate, rotate, scale ve skew)

Aslında transform özelliği ve değerleri bir kabul edilirse iki adet genel tanımımız var. Bu özellikler bize çok büyük kolaylık sağlıyor. Dönüştürme özellikleri desteğini her tarayıcı kendi ön eki ile sağlamaktadır. Ben burada tanımlamalarımı standart olan özellik ile göstereceğim, tabi örneklerde tüm tarayıcı kodlarını kullanacağım.

**transform Özelliği**

transform özelliği **none**, bir veya daha fazla transform değeri alır.

Genel kullanımı

**Yapısı :** transform: none | <dönüştürme-fonksiyonu>[<dönüştürme-fonksiyonu>]\*  
**Aldığı Değerler :** [ none | matrix | translte | translateX | translateY | scale | scaleX | scaleY | rotate | skew | skewX | skewY ]  
**Başlangıç değeri:** none  
**Uygulanabilen elementler:** tüm elementler  
**Kalıtsallık:** Yok

Aldığı değerleri dönüşüm fonksiyonları bölümünde detaylı bir şekilde inceleyeceğiz.

p{

**transform**: translate(-150px, 75px);

}

Uygulandığı paragrafı kendi orijin noktasına göre 75px sağa ve 150px aşağıya öteleyecektir. Göreceli bir tanımda(örn: % gibi) referans olarak elemanın kapsayıcısını değil kendi orijinini alacaktır.

**transform-origin Özelliği**

Dönüşümün orjin olarak nereye referans alacağını tanımlamamızı sağlar.

**Yapısı :** transform-origin: deger deger  
**Aldığı Değerler :** [ [ [ <yüzde> | <değer> | left | center | right ] [ <yüzde> |<değer>|top|center|bottom]?]<değer>?]|[[[left|center| right ] || [top | center | bottom ] ] <değer>? ]  
**Başlangıç değeri:** 50% 50%  
**Uygulanabilen elementler:** tüm elementler  
**Kalıtsallık:** Yok

Başlangıç değeri olarak 50% 50% gelir. Bu elemanın tam ortasını gösterir.(Bundan dolayı elemanı ölçeklendirdiğimizde veya döndürdüğümüzde elmanın ortasında göre işlem yapar.) Üstten 50% soldan 50%. Biz bu değeri değiştirebiliyoruz. Değer olarak CSS değerleri tanımlanabilir. İlk değeri X ekseninden mesafesini, ikinci değer Y sekseninden mesafesini tanımlar.

p{

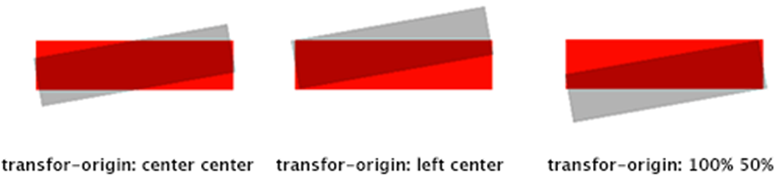
**transform-origin**:0, 0;

}

Yüzde değerlerinde referans olarak elemanın kutu alanı alınır.

Daha iyi anlamak için aşağıda da göreceğimiz döndürme(rotate) uygulayarak görelim.

transform**:** rotate**(**-10deg**);**



Yukarıda görsel olarak verilmiş örnekte kırmızı olanlar normal halleri, gri olanlar ise dönüşüm uygulanmış hallerini göstermektedir. İlk eleman başlangıç değeri olan **center center** uygulanmıştır, bu nedenle eleman mutlak konumuna göre hareket etmiştir. İkinci eleman **left center** tanımlanmıştır, bu nedenle eleman sol tarafın dikey ortasına göre dönmüştür. En son örnekte 100% 50% tanımı yapılmıştır, bu nedenle bu örnek sağ tarafa dualı dikeyde ortalı bir şekilde döndürülmüştür.

**Dönüşüm Fonksiyonları**

Dönüşüm fonksiyonlar translate(değiştir - dönüştür), rotate(döndürme), scale(ölçeklendirme) ve skew(eğme) olmak üzere 4 adettir. Birde rotate hariç bunların X ve Y için ayrı tanımlarıda var. Aşağıda detaylı incelemede göreceğiz.

**translate(değiştir - dönüştür)**

CSS3 sayesinde javascript veya flash’a gerek kalmadan animasyon veya dönüştürme işlemlerini yapabiliyoruz. Biz burada bir CSS3 transform özelliği ile nasıl sayfa içindeki elemanları taşıyacağımızı göreceğiz.

translate değeri elemanı bir noktadan başka bir noktaya taşımamızı sağlıyor.

translate **(**x**,** y**)**

Bu sayede biz elemanı orijinal konumundan x-ekseninde soldan mesafesi ve y-ekseninde üstten mesafesini tanımlayarak taşıyabiliyoruz. Basit bir kullanımı vardır.

Bir örnek yapalım;

**#tasinan**{

**transform**:translate(150px,100px);

}

**- Eksene Özgü Yöntemler**

Elemanları sadece bir eksen hareket ettirmek için translateX ve translateY özelliklerini kullanabiliriz.

Yukarıdaki örneği

**#tasinan**{

**transform**:translateX(150px);

}

aynı şekilde y ekseninde hareket için

**#tasinan**{

**transform**:translateY(100px);

}

şeklinde uygulanabilir.

**- Animasyon Yardımlı Örnek**

İleride göreceğimiz transition özelliği yardımı ile ufak bir kod ile animasyon oluşturabiliriz. Hem translate özelliğini daha iyi anlamış oluruz.

**#tasinan**{

**transition**: all 5s;

}

Fare imlecinin mavi alan üzerine getirdiğimizde animasyonlu bir şekilde kutu translate komutu ile atadığımız yeni yerine hareket edecektir. İleride CSS3 transition konusunda daha detaylı göreceğiz.

**rotate(Döndürme)**

rotate değeri yardımı ile sabit bir noktaya göre elemanı belirlenen açı ile döndürebiliyoruz. Bu sabit nokta transform-origin tanımı ile değiştirilebiliyor.

rotate**(<**açı**>)**

Bir örnek ile konuyu daha iyi anlayalım.

**.deneme**{

**transform**:rotate(45deg);

}

Yukarıdaki kod ile tanımlanan HTML elemanını 45 (deg-degrees)derece döndürdük. Saat yönüne çevirdiğimiz gibi, eksi değer vererek saat yönünün tersinede döndürme işlemi yapabiliyoruz.

Döndürme tanımlarında 4 adet değer var. deg(açı), grad(gradyen), rad(radyen) ve turn.

turn değerini biraz daha inceleyelim. turn değeri dairenin etrafını gösteriyor.

**.deneme**{

**transform**:rotate(.5trun);

}

0.5 olarakta tanımlanabilecek bu değer elemanımızı yarım daire kadar döndürecektir.

**- Animasyon Yardımlı Örnek**

Bir örnekte animasyonlu yapalım.

**.deneme**{

**transition**: all 0.5s;

}

**.deneme:hover**{

**transform**:rotate(45deg);

}

Yukarıdaki örnektede görüldüğü gibi birçok güzel uygulama bu yöntem ile yapılabilir.

**scale(Ölçekleme)**

scale değeri yardımı ile belirlene elemanın X ve Y ekseninde ölçeğini büyütebilir veya küçültebiliriz.

3 adet ölçeklendirme fonksiyonu vardır.

* scale(<sayı>[, <sayı>])
* scaleX(<sayı>)
* scaleY(<sayı>)

Bir örnek ile konuyu daha iyi anlayalım.

**.denemeBuyut**{

**transform**:scale(1.5);

}

Daha iyi bir anlatım için örnekte iki kutu kullandım birisi kutunun normal hali(mavi kutu), diğer ise kutunun ölçeklenmiş hali(kırmızı kutu). Altta kalan kutunun görünmesi için üstteki kutuya saydamlık verdim.

**- Eksene Özgü Yöntemler**

Elemanları sadece bir eksen ölçeklemek için scaleX ve scaleY özelliklerini kullanabiliriz.

Yukarıdaki örneği

**.denemeBuyut**{

**transform**:scaleX(1.5);

}

aynı şekilde y ekseninde hareket için

**.denemeBuyut**{

**transform**:scaleY(2);

}

şeklinde uygulanabilir.

**- Animasyon Yardımlı Örnek**

Bir tanede animasyonlu bir örnek yapalım.

**skew(eğme)**

skew ile eleman veya eleman gruplarına X ve Y eksenin için belirli açılar verilerek eğiklik ve yamukluk elde edebiliyoruz. 3 adet skew tanımımız vardır;

* skew( [, ])
* skewX()
* skewY()

Eğiklik tanımı yaparken tek tanım yapabildiğimiz gibi iki tanım ile x ve y için farklı açı değerleride belirleyebiliyoruz.

Bir örnek yapalım.

**.deneme**{

**transform**: skew(10deg, 20deg);

}

**- Eksene Özgü Yöntemler**

Elemanları sadece bir eksen eğmek için skewX ve skewY özelliklerini kullanabiliriz.

Yukarıdaki örneği

**.denemeBuyut**{

**transform**:skewX(10deg);

}

aynı şekilde y ekseninde eğiklik için

**.denemeBuyut**{

**transform**:skewY(20deg);

}

şeklinde uygulanabilir.

**- Animasyon Yardımlı Örnek**

Bir tanede animasyonlu bir örnek yapalım.

**Çoklu Değer Kullanımı**

Dönüşüm fonksiyonları yukarıda tanımladığımız gibi tek tek tanımlanabildiği gibi kısaltma olarak arada boşluk bırakılarak birliktede tanımlanabilir.

div {

**transform**: translate(10px 30%) scale(2.0) rotate(45deg) skew(5deg, 20deg);

}

Böylece kodlamamızda bayağı bir kısaltmaya gitmiş oluruz. Çoklu tanım uygulandığında bu fonksiyonlar soldan sağa doğru sırası ile uygulanacaktır.

**Matris Sistemi**

[w3c’nin geliştirmekte olduğu matris sistemi](http://www.w3.org/TR/SVG/coords.html#TransformMatrixDefined) ile transform tanımı yapımı karmaşık bir yapıya sahiptir. Muhtemelen kullanmayacağımız bir yapı yada belirli araçlar ile oluşturulacak bir kod yapısına sahip. Dönüştürme fonksiyonlarının her biri matriste kendine bir yer edinerek oluşturulan değerdir. Örneğin scale(1.5,1.2) tanımı matris olarak aşağıdaki şekilde gösterilir

|  |  |  |
| --- | --- | --- |
| 1.5 | 0 | 0 |
| 0 | 1.2 | 0 |
| 0 | 0 | 1 |

Bu yazımı css koduna çevirirsek;

transform**:** matrix**(**1**.**5**,** 0**,** 0**,** 1**.**2**,** 0**,** 0**)**

**Tarayıcı Destekleme Listesi** Firefox 3.5+ (-moz- öneki ile) Chrome 10+ (-webkit- öneki ile) Safari 3.2+ (-webkit- öneki ile) Opera 10.6 (-o- öneki ile) İnternet Explorer 9+ (-ms- öneki ile) **Mobil Tarayıcılar** iOS Safari 3.2+ (-webkit- öneki ile) Opera Mobile 11+ (-o- öneki ile) Android Browser 2.1+ (-webkit- öneki ile)

**Tarayıcı Desteği**

|  |  |  |
| --- | --- | --- |
| **10+ (-webkit)** | **9+ (-ms)** | **3.5+ (-moz)** |

**Mobil Tarayıcılar**

|  |  |  |
| --- | --- | --- |
| **2.1+ (-webkit)** | **3.2+ (-webkit)** | **36+** |

**TRANSITION**

**CSS Transitions**

**CSS** Transitions(geçişleri), belirli bir süre boyunca özellik değerlerini sorunsuz bir şekilde değiştirmenize olanak tanır.

Bir **CSS** geçiş efekti görmek için mouse aşağıdaki öğenin üzerine getirin:

Test

Bu bölümde aşağıdaki özellikler hakkında bilgi edineceksiniz:

·       transition

·       transition-delay

·       transition-duration

·       transition-property

·       transition-timing-function

**CSS Transition(Geçişleri) Nasıl Kullanılır?**

Bir geçiş efekti oluşturmak için iki şey belirtmelisiniz:

·       Efekt eklemek istediğiniz CSS özelliği

·       Etkinin süresi

**Not:** Süre kısmı belirtilmezse, varsayılan değer 0 olduğundan geçişin bir etkisi olmaz.

Aşağıdaki örnekte 100 piksel genişlik \* 100 piksel yükseklik ve kırmızı <div> öğesi gösterilmektedir. <div> öğesi, width özelliği için 2 saniyelik bir geçiş efekti de belirtmiştir:

Örnek:

div {

  width: 100px;

  height: 100px;

  background: red;

  transition: width 2s;

}

Geçiş efekti, belirtilen **CSS** özelliği (genişlik) değer değiştirdiğinde başlar.

Şimdi, bir kullanıcı <div> öğesinin üzerine geldiğinde width özelliği için yeni bir değer belirleyelim:

div:hover {

  width: 300px;

}

**Geçişin Hız Eğrisini Belirleme**

transition-timing-function Özelliği geçiş etkisi hız eğrisi belirtir.

Transition-timing-function Özelliği aşağıdaki değerlere sahip olabilir:

·       ease :Yavaş başlatma, sonra hızlı, ardından yavaş bitme ile bir geçiş efekti belirtir. (Bu varsayılan değerdir.)

·       linear :Baştan sona aynı hızda bir geçiş efekti belirtir.

·       ease-in :Yavaş başlatma ile bir geçiş efekti belirtir.

·       ease-out :Yavaş uçlu bir geçiş efekti belirtir.

·       ease-in-out :Yavaş başlangıç ​​ve bitişli bir geçiş efekti belirtir.

·       cubic-bezier(n,n,n,n) kübik-bezier fonksiyonunda kendi değerlerinizi tanımlamanızı sağlar.



Örnek:

div {

  width: 100px;

  height: 100px;

  background:#f39c12;

  -webkit-transition: width 2s; */\* Safari \*/*

  transition: width 2s;

}

#div1 {transition-timing-function: linear;}

#div2 {transition-timing-function: ease;}

#div3 {transition-timing-function: ease-in;}

#div4 {transition-timing-function: ease-out;}

#div5 {transition-timing-function: ease-in-out;}

div:hover {

  width: 300px;

}

Mouse ile kutuların üzerine gelin.

**Geçiş Efekti Gecikmesi**

transition-delay Özelliği geçiş etkisi (saniye olarak) bir gecikme belirtir.

Aşağıdaki örnekte başlamadan önce 1 saniye gecikme vardır:

ÖRNEK:

div {

  transition-delay: 1s;

}

**Transition(Geçiş) + Transformation**

Aşağıdaki örnek dönüşüme bir geçiş efekti ekler:

div {

  transition: width 2s, height 2s, transform 2s;

}

**CSS** geçiş özellikleri aşağıdaki gibi tek tek belirtilebilir:

div {

  transition-property: width;

  transition-duration: 2s;

  transition-timing-function: linear;

  transition-delay: 1s;

}

veya kısa gösterim özelliğini kullanarak transition:

div {

  transition: width 2s linear 1s;

}

**Solma Efekti**

Bir şeylerin solması müşterilerden oldukça yaygın bir istektir. İşlevselliği vurgulamak veya bir eylem çağrısına dikkat çekmek için harika bir yoldur.

.fade{

        width:250px;

        height:250px;

        background:#f39c12;

        transition:all 0.3s ease;

}

.fade{

        opacity:0.5;

}

.fade:hover{

        opacity:1;

}

**Renk Geçişi**

.color:hover

{

   background:#53a7ea;

}

**Büyütmek ve Küçültmek**

Bir elemanı büyütmek için genişliğini ve yüksekliğini ya da dolgusunu kullanmanız gerekiyordu. Ama şimdi büyütmek için **CSS3**'ün transform(dönüşüm) kullanabiliriz.

**Div**'inizin sınıfını "büyüyecek" şekilde ayarlayın ve ardından bu kodu stil bloğunuza ekleyin:

.test:hover

{

             -webkit-transform: scale(1.3);

    transform: scale(1.3);

}

Bir öğeyi küçültmek onu büyütmek kadar basittir. Bir öğeyi büyütmek için 1'den büyük bir değer belirleriz, küçültmek için 1'den küçük bir değer belirtiriz:

.dene:hover

{

  -webkit-transform: scale(0.8);

  transform: scale(0.8);

}

**Elemanları döndürme**

**CSS** transform(dönüşümlerin) bir dizi farklı kullanımı vardır ve en iyilerinden biri bir öğenin transform(dönüşüm özelliğine) rotate değeri verin. Div'inize rotate sınıfını verin ve **CSS**'nize aşağıdakileri ekleyin:

.rotate:hover

{

  transform: rotateZ(-30deg);

}

**Kare daire**

Şu anda gerçekten popüler bir etki, bir kare öğeyi yuvarlak bir öğeye dönüştürmek veya tam tersi. **CSS** ile, bunu başarmak basit bir etki, sadece border-radiusözelliğine ayarlıyoruz.

.circle:hover

{

        border-radius:50%;

}

**3D Gölge**

Kutuya 3d gölge verelim.

.test:hover

{

    box-shadow:

          1px 1px #53a7ea,

          2px 2px #53a7ea,

          3px 3px #53a7ea;

    transform: translateX(-3px);

}

**SHADOW**

**text-shadow**

Bu özellik bir element içindeki metne bir veya daha fazla gölge vermek için kullanılır. Aldığı değerler belli sıralaması vardır.

text-shadow değeri 4 kısımdan oluşur. Bir örnek ile açıklayalım.

**.title** {

**text-shadow**: 3px 2px 1px #000;

}

3px; İlk değer yataydaki mesafe içindir Artı değerler metnin sağından itibaren uzaklık değeridir, negatif değerler metnin soluna doğru mesafeyi gösterir.

2px; İkinci değer dikeydeki mesafe içindir. Artı değerler metnin altına doğru mesafeyi gösterir, eksi değerler ise metnin üstüne doğru mesafesini gösterir.

1px; Bulanıklık(blur) değeridir. Gölgelerin gerçeğe yaklaşması için kullanılır. Bulanık yatay ve dikey için verilen değerler göre şekil alır.

#000; renk değeri. Gölgenin renk değerini gösterir.

**CSS Öğren** CSS hakkında daha fazla bilgiye ulaşman çok kolay. CSS Dersleri sayfasında CSS hakkında bilmen gereken tüm bilgilere dair makale bulunuyor.[Hemen Öğrenmeye Başla](https://fatihhayrioglu.com/css-dersleri)

**Birden fazla gölge kullanımı**

Birden fazla gölge tanımı yapabiliyoruz. Aralarında virgül kullanarak birden fazla virgül kullanabiliyoruz. CSS2’de birden fazla gölge kullanımında üst üste binmesi durumunda normal sıralamada başta olan üstte sonradan tanımlananlar altta kalırken CSS3 ile birlikte bu durum değişti. CSS3’de tam tersi bir durum söz konusu.

color**:** #000**;**

background**:** #000**;**

text-shadow**:** 0 0 4px **#ccc,** 0 -5px 4px **#ff3,** 2px -10px 6px **#fd3,** -2px -15px 11px **#f80,** 2px -18px 18px **#f20;**

Virgül ile ayrılarak birden fazla gölge kullanılabilir.

**Tarayıcı Uyumu:** Internet Explorer desteklemiyor Firefox 3.5+ Chrome 2+ Safari 1.3+ Opera 9.5+ W3C’s CSS Level 2+ CSS Profile 2.0 ve 3.0

Çoklu gölge kullanımını Safari ve Google Chrome daha sonra desteklemiştir. Safari 4 ve Google Chrome 2’den sonra tam destek gelmiştir.

**Örnek Çalışmalar**

Konuyu daha iyi anlamak ve sonuçlarını görmek için bir kaç örnek yapalım.

p{

**font**:14px Arial, Helvetica, sans-serif;

**color**: #000;

**background**: #fff;

**text-shadow**: 2px 2px 3px #000;

}

İkinci örneğimizide gölge açısını değiştirmek için eksi değer veririz.

p{

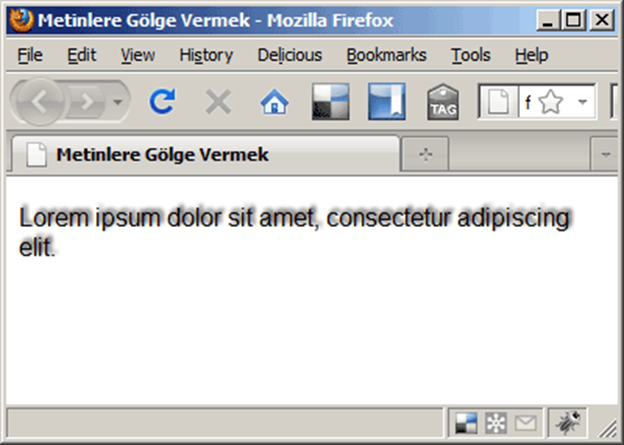
**font**:16px Arial, Helvetica, sans-serif;

**color**: #000;

**background**: #fff;

**text-shadow**: 2px -2px 3px #000;

}



Bir çok değişik denemeler ile farklı görünümler elde edebiliriz.

p{

**font**:16px Arial, Helvetica, sans-serif;

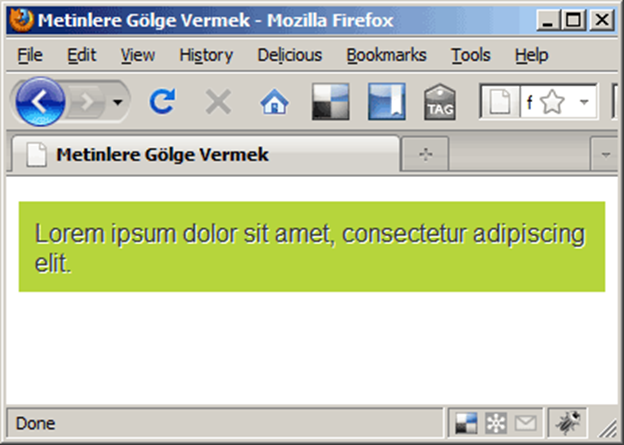
**color**: #343434;

**background**: #b6d53c;

**text-shadow**: 1px 1px #fff;

**padding**:10px

}



Gölge yardımı ile ışıltı efekti vermek.

p{

**font**:bold 16px Arial, Helvetica, sans-serif;

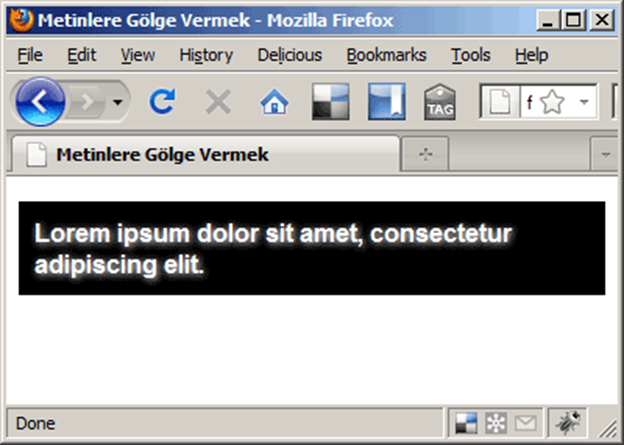
**color**: #fff;

**background**: #000;

**text-shadow**: 1px 1px 6px #fff;

**padding**:10px

}



Çoklu gölge yardımı ile ateş efekti verelim

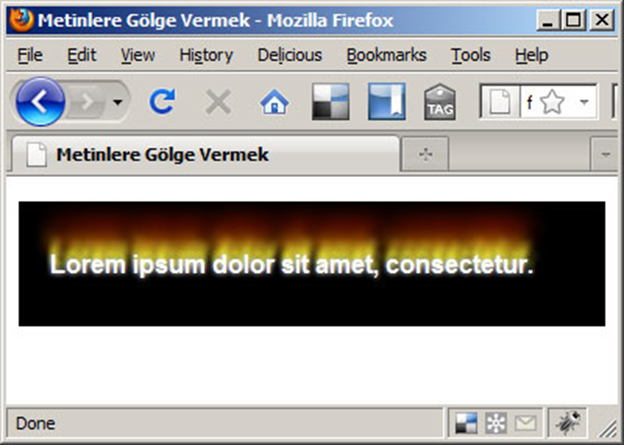
font**:bold** 16px Arial**,** Helvetica**,** sans-serif**;**

color**:** **#fff;**

background**:** #000**;**

text-shadow**:** 0 0 4px white**,** 0 -5px 4px **#FFFF33,** 2px -10px 6px **#FFDD33,** -2px -15px 11px **#FF8800,** 2px -25px 18px **#FF2200;**

padding**:**30px 20px

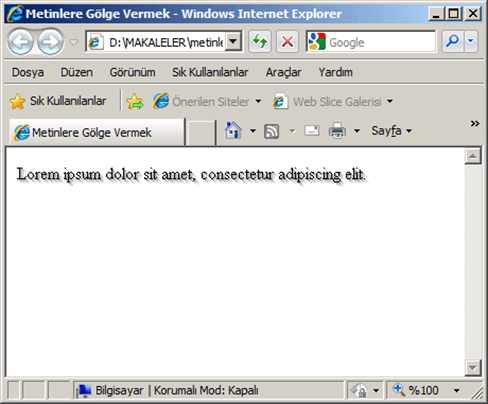


Görüldüğü gibi resim düzenleme araçları ile elde edebileceğimiz bir çok gölge efektini text-shadow ile elde edebiliriz. Farklı değerler yardımı ile bir çok gölge çeşidini elde edebiliriz.

**Shadow filtresi**

filter**:** Shadow**(**color**=**'#0000ff'**,** Direction**=**135**,** Strength**=**4**);**

Bu uygulama sonucunda aşağıdaki görüntüyü elde ederiz.



Bu özelliği kullanarak sınırlı çerçevede gölge oluşturabiliriz. Kullanacağımız yere göre bu yöntemde tercih edilebilir. Daha keskin ve belli açılarda kullanılabilir.

**Glow Filtresi**

Glow filtresi metin etrafında çevreleyen bir katman oluşturur ve görüntü olarak gölge etkisi verir.

**<!**--**[**if IE**]><**style type**=**"text/css"**>**

p**.shadowed** span {

**display**: block;

**width**: 100%;

**filter**: glow(color=#0000ff,strength=2);

}

**</**style**><![**endif**]**--**>**

Ayrıca glow ve blur filtrelerini beraber kullanarak bir çözüm üretilebilir. kilianvalkhof.com sitesinde bahsedilen bu yöntem ile gerçeğe daha yakın sonuçlar elde edilebilir.

Ayrıca aynı sitede bu yöntemi kullanan bir jquery eklentisi geliştirilmiştir. Kodlamayı kolaylaştıran bu yöntemde denenebilir. Ayrıntılı bilgi için tıklayınız.

**Tekrarlayan Metin Kopyası ile Gölge Oluşturma**

Aynı metni iki kere veya daha fazla kopyalayıp position ve konumlandırma değerleri ile gölge etkisi verilen bir yöntemdir. Farklı tarayıcılarda aynı etkiyi verir. Sorun oluşturan tek yanı ise içerikte yazının birden fazla geçmesi ve fazladan etiket eklemesidir.

Kodlaması çok basittir.

<div class="baslik">

    <h1>Lorem ipsum dolor sit amet consectetuer</h1>

    <h6>Lorem ipsum dolor sit amet consectetuer</h6>

</div>

CSS kodu

**.baslik** {

**position**:relative;

**font-family**:arial;

}

**.baslik** h1 {

**position**:absolute;

**top**:2px;

**left**:2px;

**font-size**:30px;

**color**:#C9D8E9;

**padding**:0;

**margin**:0;

}

**.baslik** h6 {

**position**:absolute;

**top**:0;

**left**:0;

**font-size**:30px;

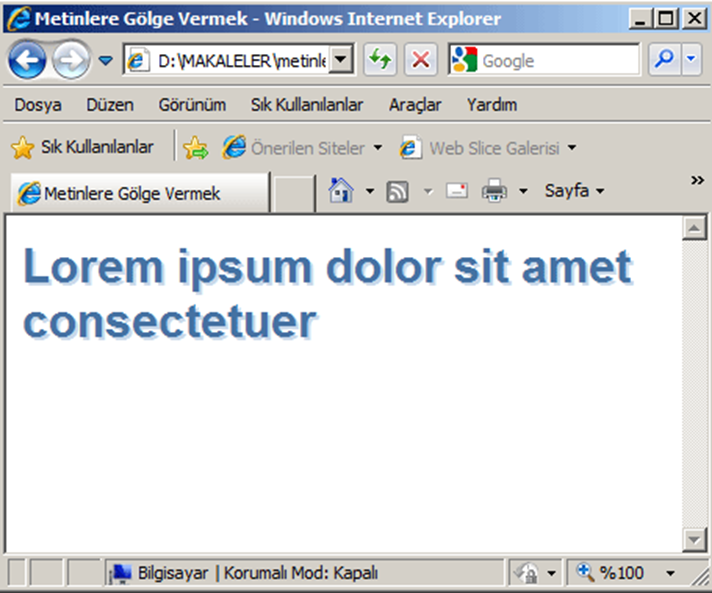
**color**:#4471A2;

**padding**:0;

**margin**:0;

}

Sonuç aşağıda göründüğü gibi olur.



**BOX SHADOW**

**Klasik Gölge (Drop Shadow)**



* CSS Kodu

|  |
| --- |
| div{ |

|  |
| --- |
| -moz-box-shadow:5px 5px 15px 3px #666; |

|  |
| --- |
| -webkit-box-shadow: 5px 5px 15px 3px #666; |

|  |
| --- |
| box-shadow: 5px 5px 15px 3px #666; |

|  |
| --- |
| } |

* HTML Kodu

|  |
| --- |
| <**div**>Klasik Gölge</**div**> |

**Dahili Gölge (Inner Shadow)**  
****

* CSS Kodu

|  |
| --- |
| div { |

|  |
| --- |
| -moz-box-shadow:inset 0 0 20px #000000; |

|  |
| --- |
| -webkit-box-shadow: inset 0 0 20px #000000; |

|  |
| --- |
| box-shadow: inset 0 0 20px #000000; |

|  |
| --- |
| } |

* HTML Kodu

|  |
| --- |
| <**div**>Dahili Gölge</**div**> |

**Tek Taraflı Gölge (One Side Shadow)**  
  


* CSS Kodu

|  |
| --- |
| div { |

|  |
| --- |
| -webkit-box-shadow: 0 11px 8px -6px #666; |

|  |
| --- |
| -moz-box-shadow: 0 11px 8px -6px #666; |

|  |
| --- |
| box-shadow: 0 11px 8px -6px #666; |

|  |
| --- |
| } |

* HTML Kodu

|  |
| --- |
| <**div**>Tek Taraflı Gölge</**div**> |

**Dışa Dağılım (Outer Glow)**  
  


* CSS Kodu

|  |
| --- |
| div { |

|  |
| --- |
| -webkit-box-shadow: 0 0 6px 4px #666; |

|  |
| --- |
| -moz-box-shadow: 0 0 6px 4px #666; |

|  |
| --- |
| box-shadow: 0 0 6px 4px #666; |

|  |
| --- |
| } |

* HTML Kodu

|  |
| --- |
| <**div**>Dışa Dağılım</**div**> |