**1. Fundamentals**

**CSS NEDİR?**

İngilizcesi **“Cascading Style Sheets”** olan CSS açılımı, **“Basamaklanmış Stil Katmanları”**anlamına gelmektedir. **“CSS nedir?”** sorusunun cevabı ise web sitelerinin görsel olarak şekillendirilmesine olanak tanıyan ve kendine has kuralları olan bir tanım dilidir şeklinde ifade edilebilir. **HTML ve JavaScript ile birlikte en temel web teknolojileri** arasında bulunan CSS, web sayfaları üzerinde oldukça fazla görsel denetim sunar.

CSS sayesinde web sayfalarının mizanpajı, renkleri, kullanılan fontlar, ara başlıklar, görsel efektler ve diğer görsel unsurlar üzerinde etkili ve fonksiyonel bir kontrol sağlayabilirsiniz. Kullanıcı dostu olan ve görsel açıdan etkileyici web siteleri için HTML ve JavaScript’in yanı sıra yazılan CSS kodları son derece önem taşır. **Etkili bir CSS kodlaması, web sitelerine yüksek bir kullanılabilirlik sunar.**

CSS, web sayfalarının tasarlanmasını oldukça kolaylaştıran ve büyük kolaylıklar sağlayan **esnekliği** ile de ön plana çıkar. Bu özellik sayesinde web sayfalarınızın tümünde ya da sadece belirli bölümlerinde yapacağınız güncellemeleri çok kısa süre içinde gerçekleştirebilirsiniz. CSS ile bir web sitesinde bulunan bütün sayfaların ya da sadece istenilen sayfanın görsel özellikleri **oldukça kolay bir şekilde kontrol edilebilir.**

**CSS Sürümleri ve Özellikleri Nelerdir?**

CSS’in teknik olarak sahip olduğu özellikler, açılımı **World Wide Web Konsorsiyumu** olan **W3C** kuruluşu tarafından belirlenmektedir. W3C içerisinde **“CSS Çalışma Grubu”** isminde bir grup bulunur ve CSS’in teknik özellikleri ile ilgili çeşitli çalışmalar bu grup tarafından gerçekleştirilir. CSS Çalışma Grubu tarafından yapılan çalışmalar W3C üyeleri tarafından tartışılır ve resmi olarak onaylandıktan sonra ilgili yenilikler uygulanır.

Her bir yeni CSS sürümü kendisinden önceki sürüm üzerine inşa edilir ve **CSS 1, CSS 2, CSS 3** gibi sürüm numaralarıyla adlandırılır. İlk CSS sürümü 17 Aralık 1996’da W3C tarafından yayınlanan **CSS 1**‘dir. CSS 1; yazı tipleri, metinler, arka planlar, tablolar ve kenarlıklar gibi web sayfası ögeleri ile ilgili temel denetimleri içeren bir sürümdür.

CSS 1’in bir üst sürümü olan **CSS 2,** 1998 yılında yayınlanmıştır ve daha fonksiyonel web sitelerinin tasarlanabilmesi için çeşitli yeniliklere sahiptir. Gelişmiş konumlandırma seçenekleri, Z-index, işitsel stil sayfaları desteği, çift yönlü metin ve gölgelendirme gibi çeşitli gelişmiş özellikleri vardır. **CSS 2.1** sürümü ile CSS 2’de bulunan hatalar giderilmesi sağlanmış ve **daha stabil bir çalışma sağlanmıştır.**

W3C tarafından Haziran 1998’da yayınlanan **CSS 3**, CSS’in en son sürümüdür. Önceki sürümlerin üzerine inşa edilen CSS 3; seçiciler, renkler, isim alanları ve medya sorguları gibi konularda yeni özellikler getirmiştir. Yuvarlak köşeler, geçişler, gölgeler, animasyonlar ve degradeler gibi birçok CSS 3 özelliği sayesinde **daha görsel ve kullanıcı deneyimini ön plana çıkaran gelişmiş web sayfaları tasarlayabilirsiniz.**

**CSS Kullanım Şekilleri Nelerdir?**

**CSS yazılımı**web sayfalarına ait özelliklerin tek tek veya toplu olarak kontrol edilmesini sağlayan oldukça esnek ve kullanışlı bir yapıya sahiptir. CSS’de; **harici stil sayfası, dahili stil sayfası ve etiket içine yazılan stiller** olmak üzere 3 farklı yöntemle web sayfalarınıza kodlarınızı ekleyebilirsiniz. **Her bir yöntemin kullanım amacına ve yerine göre sahip olduğu çeşitli avantajlar vardır.**

Sadece bir CSS sayfası ile birden çok web sayfanızın stil özelliklerini kontrol etmek istiyorsanız harici stil sayfası yöntemi ile**CSS kodu**ekleyebilirsiniz. Tek bir web sayfasında yer alan ögeleri kontrol etmek ve gereken biçimlendirmeleri yapmak için dahili stil ekleme metodunu kullanabilirsiniz. Etiket içine yazılan kodlar ise sadece ilgili etiketi kontrol eder ve gerekmedikçe kullanılması önerilmez.

**CSS Kodlama ile Neler Yapılabilir?**

Yaygın inanışın aksine, yazılan CSS kodları sadece web sitelerini daha görsel ve estetik bir görünüme kavuşturmak için kullanılmaz. CSS’in web sitesinin ziyaretçilerine daha üst düzey bir kullanıcı deneyimi yaşatmaya olanak tanıyan birçok güçlü özellikleri vardır. Web sayfalarında sadece CSS ile yapılabileceğiniz işlemlerin dışında; **CSS, HTML ve JavaScript dilinin gücünü birleştirerek oldukça etkileyici sonuçlar elde edebilirsiniz.**

Yazı boyutlandırma, farklı başlık stilleri, kenar süsleri, gölgelendirme, yatay ve dikey menüler, sayfa mizanpajı ve renklerle ilgili işlemler gibi birçok işlem sadece CSS kodları kullanılarak gerçekleştirilebilir. **CSS kodları web sayfalarında temel düzeydeki geçiş efektleri ve basit animasyonlar** gibi çeşitli işlemlere de izin verir. Tablo ve form elemanlarına yazdığınız kodlar ile sitelerinizi daha etkileşimli hale getirebilirsiniz.

**CSS, HTML ve JavaScript’in** sahip olduğu potansiyelin harmanlanması sayesinde oluşturulan birçok üst düzey geçiş efekti, animasyon ve menü tasarımları ile ziyaretçilere daha gelişmiş bir web deneyimi yaşatılabilir. **HTML5 ve CSS’in** son sürümü olan CSS 3’ün birlikte etkin kullanımı ile oldukça etkileyici web tasarımları gerçekleştirilebilirsiniz. Daha gelişmiş web animasyonları için çeşitli **JavaScript**kütüphanelerinden de faydalanabilirsiniz.

**CSS Yazılar**

Bu sayfada HTML sayfanızda kullandığınız yazıların değiştirebileceğiniz belli başlı özellikleri (renk, hiza, kalın vb) hakkında bilgi verilecektir.  
  
Yazı biçimlendirme ile ilgili komutlardan tanıyacaklarımız:

* color
* text-align
* text-decoration
* text-transform
* text-indent

**color: Yazı Rengi**

Yazının rengini belirlemenizi sağlar 

      p { color: blue; }

**text-align: Hizalama**

Yazının yatay yönde ne şekilde hizalanacağını belirtmenizi sağlar. En çok kullanılan dört hizalama yöntemi:

* left : sol
* right : sağ
* center : ortalanmış
* justify : iki yana yasla

Aşağıda paragrafların iki yana yaslanacağını belirttik:

      p { text-align: justify; }

**text-decoration: Yazı Biçimi**

Yazının biçiminde bazı değişiklikler yapmamızı sağlar. Örneğin tarayıcı varsayılanında sürekli A elementlerindeki altı çizgili dekorasyonu kaldırmak için aşağıdaki yazılabilir.

      a { text-decoration: none; }

**text-transform: Yazıda Değişiklik Yap**

Yazılan bir yazıyı otomatik olarak tüm harflerini büyük harfle yazmayı ya da tamamını küçük harfle yazmayı sağlar.

* uppercase : tüm harfleri büyük harf yap
* lowercase : tüm harfleri küçük harf yap

      h1 { text-transform: upperline; }

**text-indent: Soldan Boşluk**

Bir yazının başında belirtilen büyüklükte boşluk bırakmanızı sağlar. Özellikle paragraflar için kullanılır.

      /\* 50 piksel ilk satırda boşluk bırakılacak \*/

      p { text-indent: 50px; }

**NOT:** Piksel (px), inç (in), punto (pt), santim (cm), yüzde (%) gibi stillerin sonuna eklenen ölçüler her zaman sayıya bitişik yazılır. 50px doğru ancak 50 px yanlış.

**Diğer Yazı Şekillendirme Komutları**

**line-height:** Satır yüksekliğini belirtmemizi sağlar.  
  
**letter-spacing:** Harfler arasında belirtilen değerde boşluk koyar.  
  
**word-spacing:** Kelimeler arasında belirtilen değerde boşluk koyar.

**CSS Yazı Tipleri**

Bu sayfada HTML sayfanızda kullandığınız yazıların yazı tipi özelliklerini nasıl değiştirebileceğiniz hakkında bilgi verilecektir.  
  
Yazı tiplerini biçimlendirirken en sık kullandığımız komutlar:

* font-family
* font-size
* font-style
* font-weight

**font-family: Yazı Tipi Belirlemek**

Yazı tipleri klasörümüzü açtığımızda birçok yazı tipi görürüz. Bunları HTML sayfamızda kullanmak için adını belirtmemiz gerekir. İşte bu kod yazı tipi seçmemizde bize yardımcı olacaktır.

      h1 { font-family: Times New Roman; }

      p { font-family: Times New Roman, Arial, Helvatica; }

      a { font-family: Sans-serif; }

Yukarıdaki birinci örnekte sadece tek yazı tipi adı belirttik (Times New Roman), ancak bir sonrakinde aralarına virgül koyarak daha fazla belirttik. Bu tarz kullanımlarda öncelikle dikkate alınan ilk verdiğimiz yazı tipi olacaktır. Eğer verdiğimiz yazı tipi o kullanıcıda yoksa bir sonraki ile görüntülenecektir.

**font-size: Yazının Büyüklüğü (Puntosu)**

Yazı büyüklüğünü font-size ile belirtiriz. Ölçü birimi olarak aşağıdakileri kullanabiliriz:

* % : yüzde olarak belirtmek
* in : inç olarak belirtmek
* cm : santim olarak belirtmek
* mm : milimetre olarak belirtmek
* em : varsayılan yazı büyüklüğüne göre oranlamak
* pt : punto olarak belirtmek
* px : piksel olarak belirtmek

Örneğin yazımızın 12 punto olması için yazmamız gereken komut:

      h2 { font-size: 12pt; }

14 piksellik bir yüksekliğe sahip yazı istiyorsak:

      h2 { font-size: 14px; }

Varsayılan yazı tipinin %150 fazlasını EM ile belirtirsek:

      h2 { font-size: 1.5em; }

Yukarıdaki örneği yüzde ile ifade etmek isteseydik:

      h2 { font-size: 150%; }

**font-style: İtalik (Sağa Yatık) Yazmak**

Yazımızın sağa yatık (italic) olması için aşağıdaki kodu kullanabiliriz.

      h2 { font-style: italic; }

Eğer italik bir yazıyı iptal edeceksek aşağıdaki kodu yazmalıyız:

      h2 { font-style: normal; }

**font-weight: Kalın Yazmak**

Yazımızın kalın (bold) olmasını istiyorsak aşağıdaki kodu kullanmalıyız.

      h2 { font-weight: bold; }

Eğer kalın bir yazıyı normal haline dönüştüreceksek aşağıdaki kodu yazmalıyız:

      h2 { font-weight: normal; }

CSS'de Renk İfadeleri

CSS dosyamızda yer yer renkleri belirtmemiz gerekebilir. Bunun için çeşitli yollar var. Bunlardan en çok kullanılan üç tanesi:

* Onaltılık (Hex) Renkler
* RGB (Kırmızı, Yeşil, Mavi) Renkler
* Tarayıcı Renk İsimleri

**Onaltılık (Hex) Renkler:** #000000 şeklinde önce diyez sonra 6 adet 0-9 ve A-F değerleri alabilen sembollerden oluşur. Örneğin #404040 bir rengi ifade etmektedir.  
  
Bu renk kodlarına resim işlem programlarınız yardımıyla (photoshop, paint shop pro gibi) ulaşabilirsiniz.  
  
**RGB (Kırmızı, Yeşil, Mavi) Renkler:** Bir renk oluştururken kırmızı, yeşil ve mavinin tonlarını belirterek renk oluşturmamızı sağlar. Örneğin **rgb( 0, 0, 0 )** siyah rengi belirtir. Mavi bir renk elde etmek istediğimizde Red - Green - Blue üçlemesinde en sonda yer alan mavinin değerini arttırmamız yeterli olur. Yani rgb( 0, 0, 255 ) mavi rengi verir. Windows'daki Paint Brush (MS Paint) programı ve diğer resim editörleri yardımıyla RGB değerlerini alarak renk oluşturabilirsiniz.  
  
**Tarayıcı Renk İsimleri:** İngilizce olarak belirlenmiş bazı renk adlarını yazarak renk elde edebilirsiniz. Örneğin white beyaz, black siyah, blue mavi rengini verecektir.  
  
Aşağıda renklerle ilgili örnek bir kullanım var:

      p { color: #CC0000; }         /\* Koyu kırmızı \*/

      div { color: line; }        /\* Yeşilin bir tonu \*/

      code { color: rgb( 0, 0, 255 ); }       /\* Mavi renk \*/

background-color: Arkaplan Rengi

Bir elementin arkaplan rengini belirlememizi sağlar.

      div { background-color: #EFEFEF; }

**SEÇİCİLER**

**Id , etiket ismi ve class** olarak etiketlere ulaşabileceğimiz 3 temel yöntemden başka kullanabileceğimiz belli amaçlara göre hazırlanmış Selector türleri vardır.

Bu selector türlerini şu şekilde guruplayabiliriz ;

* Group Selectors
* Attribute Selectors
* Pseudo Selectors

**Css' de Group Selector Nedir ve Nasıl Kullanılır ?**

**Css Group Selector**' larını **bir kapsam içinde aradığımız etiketler** için kullanıyoruz. Örneğin farklı seviyelerde olan bir **<p>** etiketini düşünelim ;

<**div**>

   <**p** id="first"> </**p**>

   <**p** class="description"> </**p**>

</**div**>

<**p**> </**p**>

ve bu etiketlerden **sadece div altındaki p etiketlerine** ulaşmak istiyorsak bu durumda **group selectors'** ları kullanmamız gerekiyor.

**div** **p** {

   /\* div altındaki tüm p etiketleri \*/

}

**div** #first{

   /\* div altında id' si first olan eleman \*/

}

**div** .description{

   /\* div altında class' ı description olan eleman \*/

}

**Diğer group selector kullanımları ;**

**1 :** (\*)     => tüm etiketler   
**2 :** (p)    => tüm p etiketleri  
**3 :**(div p) => div içindeki tüm p etiketleri  
**4 :**(div,p) => tüm div ve tüm p etiketleri  
**5 :**(div > p) => üst etiketi div olan tüm p etiketleri  
**6 :** (p ~ div)=> p ile aynı seviyede tüm div etiketleri  
**7 :** (p + div)=> p etiketinden sonra gelen aynı seviyedeki div etiketi

**Css' de Attribute Selector Nedir ve Nasıl Kullanılır ?**

**Attribute Selector**' larını etiketlerde belirtilen **attribute**özelliğine göre kullanıyoruz.

Örneğin ;

<**img** src="resim.jpg" alt="doğa resimleri">

img etiketi için **src ve alt**özelliği bir **attribute özelliğidir** ve bu img etiketine ulaşabilmek için**bu attribute özelliklerini kullanabiliriz.**

Eğer sayfadaki **alt attribute olan tüm img** etiketlerine ulaşmak istersek ;

1-) Attribute ismine göre;

     \*\* alt attribute özelliğine sahip olan tüm img etiketleri seçilir.

**img**[alt] {

}

2-) Attribute özelliğine atanan değere göre ;

  \*\* alt attribute özelliğine "doğa resimleri" atanmış tüm img etiketleri seçilir.

**img**[alt="doğa resimleri"] {

}

3-) Attribute özelliğine atanan değere göre ancak bu değer içinde geçen belli bir kelime aranır.

   \*\* alt attribute özelliği içinde doğa kelimesi geçen tüm img etiketleri seçilir.

**img**[alt~="doğa"] {

}

4-) Attribute özelliğine atanan değere göre ancak bu değer içinde geçen belli bir kelimenin en başta olması aranır.

  \*\* alt attribute özelliği doğa kelimesi ile başlayan tüm img etiketleri seçilir.

**img**[alt^="doğa"] {

}

5-) Attribute özelliğine atanan değere göre ancak bu değer içinde geçen belli bir kelimenin en sonda olması aranır.

  \*\* alt attribute özelliği doğa kelimesi ile biten tüm img etiketleri seçilir.

**img**[alt$="doğa"] {

}

**Css' de Pseudo Selectors Nedir ve Nasıl Kullanılır ?**

Pseudo Selectors' lar 2 grup altında ele alınıyor.

* Pseudo Classes
* Pseudo Elements

**Pseudo Class Selector' ları** html etiketlerinin durumunu kontrol ettikten sonra uygulayacağımız css kodları için kullanılır.

Kullanımı şu şekildedir

**selector**:pseudo-class {

**property**:value;

}

**<a>** etiketi için kullanabileceğimiz pseudo class selector kullanımları ;

/\* henüz ziyaret edilmemiş link \*/

**a**:link {

**color**: blue;

}

/\* ziyaret edilmiş link \*/

**a**:visited {

**color**: gray;

}

/\* mouse link üzerindeyken \*/

**a**:hover {

**color**: yellow;

}

/\* linke tıklanma anı \*/

**a**:active {

**color**: red;

}

Yukarıda görüldüğü gibi tanımlanan bir link için kullanabileceğimiz bazı **pseudo class selector '**ları vardır. Tabi ki bu selector'lar sadece **<a>** etiketi için değildir. Örneğin bir **div etiketi için hover** olayını gerçekleştirebiliriz ;

**div**:hover {

**background-color**: blue;

}

div' in üzerine mouse ile geldiğimiz an da div' in arkaplanı mavi olur.

Ya da bir **div içerisindeki bir p etiketini** istediğimiz zaman **görünür hale** getirebiliriz

<**div**>

    p etiketini görmek için mouse ile üzerine gel

   <**p**>Paragraf</**p**>

</**div**>

<**style**>

**p** {

**display**: none;

**background-color**: yellow;

**padding**: 20px;

   }

**div**:hover **p** {

**display**: block;

   }

</**style**>

**\*\* display : none; =>** ulaştığımız bir etiketin görünmez olmasını sağlar. Örneğimizde **<p>** etiketi içindeki **"Paragraf"** yazısı **görünmez**olur. Ancak **pseudo class selector**' ını kullanarak **<div> etiketi üzerine mouse ile geldiğimiz anda** yani <div> etiketi için **hover**olayı gerçekleştiği anda **<p> etiketinin display özelliğini block yaparak görünür hale getirmiş oluruz.**

**\*\* first-child ve last-child pseudo class selector kullanımı:**Bir kapsam içindeki ilk alt eleman ya da son alt eleman seçiminde kullanılır.

**p**:first-child {

**color**: blue;

}

Buradaki örnekte her hangi bir kapsam içindeki ilk <p> etiketi seçilir.

**p** **i**:first-child {

**color**: blue;

}

Buradaki örnekte tüm <p> etiketleri içindeki ilk <i> etiketi seçilir.

**Pseudo Element Selector' ları** ise bir **etiketinin her hangi bir parçasını seçmek** istediğimizde kullanırız. Pseudo class selector' larından farklı olarak **"::" sembolü**kullanılır.

Kullanımı şu şekildedir ;

**selector**::pseudo-element {

**property**:value;

}

Örneğin; **<p>** etiketinin kapsamındaki yazı içinde ilk satırı seçmek istersek **::first-line** özelliğini kullanabiliriz.

**p**::first-line {

**color**: red;

}

Ya da  **<p>** etiketinin kapsamındaki yazı içinde ilk harfi seçmek istersek **::first-letter**özelliğini kullanabiliriz.

**p**::first-letter {

**color**: green;

**font-size**: xx-large;

}

Bu örneğimizde paragrafın ilk harfi yeşil ve büyük font olarak karşımıza gelir.

Sıklıkla kullandığımız **::after ve ::before pseudo element selector**' ları ise bir etiketten **hemen önce ya da sonra bir içerik eklemek** istediğimizde kullanılır.

**p**::after {

**content**: " - eklenecek yazı";

}

\*\* <p> etiketinin bittiği yerden itibaren content içindeki yazı etiket sonuna eklenir. Aynı şekilde <p> etiketinden hemen önce eklemek istersek **:before pseudo element selector'**ını kullanabiliriz.

\*\* Peki **::after ve ::before pseudo element selector'**ları sadece bu amaç içinmi kullanırız ?

.clearfix::after {

**content**: "";

**display**: block;

**clear**: both;

}

şeklinde sıklıkla kullandığımız css kod parçasını float özelliğini kullandıktan sonra clear işlemi yapmak için kullanıyoruz.

**LİNK STİLLERİ**

**Link <a>** öğesi olarak HTML’de kullandığımız bir **bağlantı** elemanıdır. Başka bir web sayfasına veya başka bir yere **köprü** oluşturmak için kullanılır. Temel etiketlerden olan **<a>** öğesini CSS kullanarak çok *farklı* biçimlerde gösterebiliriz.

CSS ile bir **link** düzenlemek isteseniz yapmanız gereken CSS seçicisi ile **<a>**öğesini seçmektir.

<style>

    a {   color: hotpink;

    }

</style>

<p><b><a href="yazilimkodlama.com" target="\_blank">CSS Örnekleri</a></b></p>

Linkler *dört* farklı duruma sahiptir. Bunlar;

·       **a:link**  – normal, ziyaret edilmemiş bir bağlantı görünümü

·       **a:visited**  – kullanıcının ziyaret ettiği bir bağlantı görünümü

·       **a:hover** – kullanıcı mouse imlecini üzerine getirdiğinde oluşan bağlantı görünümü

·       **a:active** – tıklandığı anda oluşan bağlantı görünümü

Bu dört farklı durumu **<a>** öğesi için ayrı ayrı biçimleyebilirsiniz.

<style>

    /\* ziyaret edilmemiş \*/

    a:link {  color: red;  }

    /\* ziyaret edilmiş \*/

    a:visited {  color: green;  }

    /\* mouse üzerinde \*/

    a:hover {  color: darkcyan; }

     /\* seçilmiş link \*/

    a:active {  color: blue;  }

</style>

 <p><b><a href="yazilimbilisim.net" target="\_blank">CSS Örnekleri</a></b></p>

**Not :** **<a>** öğesinin farklı durumlarını biçimlerken mutlaka yukarıdaki sıralamaya uyulmalıdır.

**Link Dekorasyonu**

**<a>** Öğesini biçimlemeden kullanıldığında altı çizgili olduğu görülür. Bu onun bir **link** olduğunu ve diğer metinlerden *ayrı* görünmesinin gereği olarak sunulmuştur. Bu iyi bir görüş olsa da bu türden bir görünüm yerine daha farklı biçime elbette **CSS** ile getirmek mümkündür. Böyle bir durumda alt çizgisinden kurtulmak isteyebilirsiniz.  **text-deocoration** Özelliğinin **none** değerini kullanılarak <a> etiketinin standart **altı çizgisini yok edebilirsiniz.**

<style>

    a:link {   text-decoration: none;   }

    a:visited {  text-decoration: none;  }

    a:hover {  text-decoration: underline;  }

    a:active {  text-decoration: underline;  }

</style>

<p><b><a href="yazilimkodlama.com" target="\_blank">CSS Örnekleri</a></b></p>

**CSS Listeler (Lists)**

HTML'de iki ana liste türü vardır:

* sırasız listeler **(<ul>)** - liste öğeleri madde işaretleri ile işaretlenmiştir.
* sıralı listeler **(<ol>)** - liste öğeleri sayılar veya harflerle işaretlenmiştir.

CSS liste özellikleri şunları yapmanızı sağlar:

* Sıralı listeler için farklı liste öğesi işaretçileri ayarlama.
* Sırasız listeler için farklı liste öğesi işaretçileri ayarlama.
* Liste öğesi işaretçisi olarak bir resim ayarlama.
* Listelere ve liste öğelerine arka plan renkleri ekleme.

**Farklı Liste Öğesi İşaretçileri**

**list-style-type** özelliği, liste öğesi işaretçisinin türünü belirtir.

Aşağıdaki örnek, mevcut liste öğesi işaretlerinden bazılarını göstermektedir.

**Örnek:**

ul.a {

  list-style-type: circle;

}

ul.b {

  list-style-type: square;

}

ol.c {

  list-style-type: upper-roman;

}

ol.d {

  list-style-type: lower-alpha;

}

**Not:** Değerlerin bazıları sırasız listeler, bazıları ise sıralı listeler içindir.

**Liste Öğesi İşaretleyicisi Olarak Bir Görüntü Ekleme**

**list-style-image** özelliği, bir resmi liste öğesi işaretçisi olarak ayarlar.

**Örnek:**

ul {

  list-style-image: url('profil.gif');

}

**Liste Öğesi İşaretçilerini Konumlandırma**

**list-style-position** özelliği, liste öğesi işaretçilerinin konumunu belirtir.

**Örnek:**

ul.a {

  list-style-position: outside;

}

ul.b {

  list-style-position: inside;

}

**"list-style-position: outside;"**madde işaretlerinin liste öğesinin dışında olacağı anlamına gelir. Liste öğesinin her satırının başlangıcı dikey olarak hizalanır.

* Kahve - Sıcak bir içecektir.
* Çay
* Kola

**"list-style-position: inside;"** madde işaretlerinin liste öğesinin içinde olacağı anlamına gelir. Liste öğesinin bir parçası olduğu için, metnin bir parçası olacak algılanır ve metni sağa doğru iter.

* Kahve - Sıcak bir içecektir.
* Çay
* Kola

**Varsayılan Ayarları Kaldırma**

**list-style-type:none** özelliği, işaretleri/maddeleri kaldırmak için de kullanılabilir. Listenin ayrıca varsayılan kenar boşluğa (margin) ve iç boşluğa (padding) sahip olduğunu unutmayın. Bunu kaldırmak için <ul> veya <ol>'a margin:0 ve padding:0 eklemelisiniz.

**Örnek:**

ul {

  list-style-type: none;

  margin: 0;

  padding: 0;

}

**Liste - Steno (Kısaltma) Özelliği**

Liste stili özelliği bir stenografi özelliğidir. Tüm liste özelliklerini tek bir satır şeklinde yazılabilir.

**Örnek:**

ul {

  list-style: square inside url("profil.gif");

}

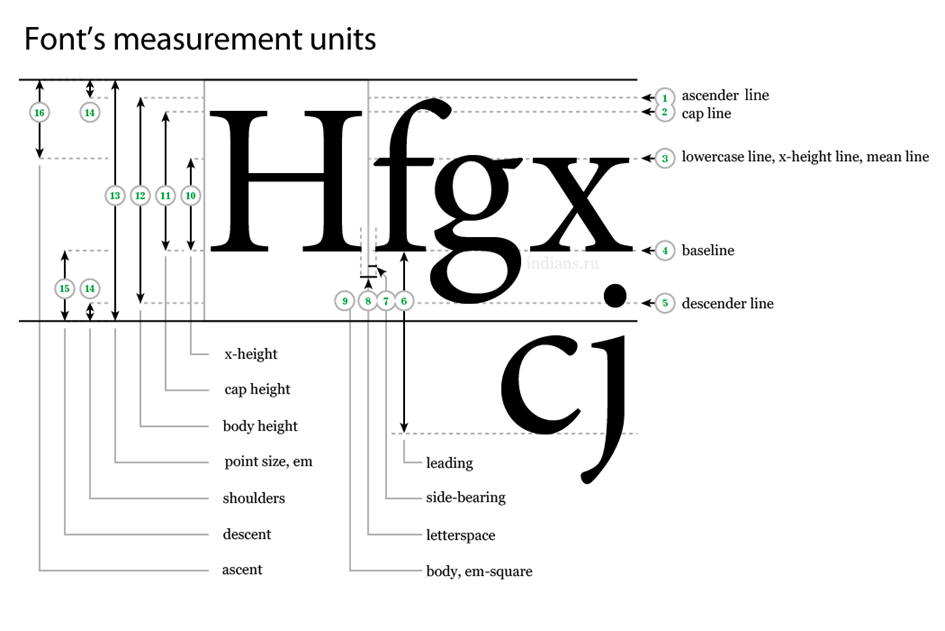
Steno özelliğini kullanırken, özellik değerlerinin sırası dikkat edilmelidir.

* **list-style-type** (bir liste stili resim belirtilmişse, resim herhangi bir nedenle görüntülenemiyorsa bu özelliğin değeri görüntülenecektir)
* **list-style-position** (liste öğesi işaretçilerinin içerik akışının içinde mi yoksa dışında mı görüneceğini belirtir)
* **list-style-image** (bir resmi liste öğesi işaretçisi olarak belirtir)

**CSS’ de Ölçü ve Uzunluk Birimleri**

CSS içinde **uzunluk tanımlamaları** için birden fazla ölçü birimi kullanılmaktadır. Bunların bazıları matbaacılıktan, bazıları günlük hayatımızda kullandığımız ölçülerden gelirken bazıları ise bilişim teknolojilerinde özellikle **web sitelerinin**tasarlanmasında ve ölçülmesinde önemli role sahip birimlerdir.

Tüm CSS uzunluk birimleri, **pozitif**ya da**negatif** sayılar olarak ifade edilebilir, ancak bazı özellikler sadece pozitif sayıları geçerli görür (örneğin yazı tipi boyutu).



CSS de ölçü birimlerini genel olarak 2 kısıma ayırabiliriz:

1.    Mutlak(Absolute) Ölçü Birimleri

2.    Göreceli (Relative) Ölçü Birimleri:

·       Font Göreli Uzunluklar

·       Pencere Boyu veya Bakış Alanı Birimi (Viewport)

Tasarım yaparken çıkış ortamı göz önüne alınıp(ekran,baskı vb.) uygun birimlerin seçilmesi ise buradaki kilit nokta diyebiliriz. Yanlış bir kullanımda tasarımımızı efektif bir şekilde çıktıya yansıtamayabiliriz.

**1) MUTLAK (ABSOLUTE) ÖLÇÜ BİRİMLERİ**

Mutlak ölçü birimleri **(cm, mm, in, pt ,pc ve px\*)** çıkış ortamı **baskı** olacak belgelerde kullanışlı ölçülerdir. Web tasarımı için **uygun değildir** (pixel istisnai bir durum). Belirli bir birimin aynı değerlerinin mutlak uzunlukları farklı ekranlarda farklılık gösterebilir. Bu, ekran **DPI** (inç başına nokta sayısı) farklılığından kaynaklanır. Daha yüksek çözünürlüklü ekranlar, daha küçük çözünürlüklü ekranlara kıyasla **daha yüksek bir DPI** değerine sahiptir ve bu nedenle görüntü veya metin daha **küçük**görünür.Bu sebeple, belgenin tamamını görmek için kullanıcının sürekli ekranı kaydırması gerekecektir. Bu da iyi bir kullanıcı deneyimi oluşturmaz.

·       cm (santimetre)

·       mm (milimetre)

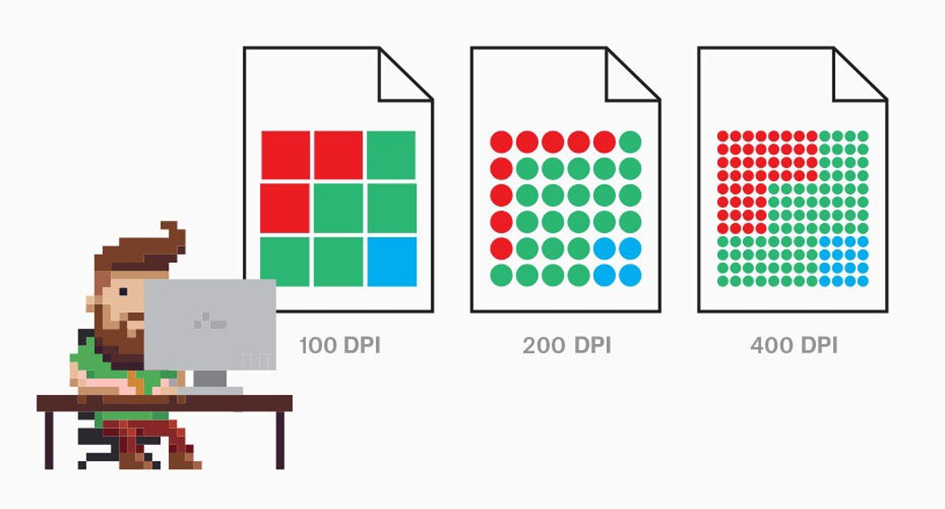
·       in (inç) / (1in = 96px = 2.54cm)

·       pc (picas) / (1pc = 12 pt)

·       pt (point) / (1pt = 1in’in 1/72’si)

·       px\* (piksel) / (1px = 1/96th of 1in)

**Piksel (px)** web tasarımında çokça tercih edilen bir birimdir. Pixel ile ilgili istisnai durumu açıklamak gerekirse, px cihazın görüntüleme biçimine göre değerlendirilir. Bu anlamda, low-dpi değerine sahip bir cihazda bir CSS pikseli (px) bir cihaz pikseline (dot) eşit diyebiliriz. Bu tarz bilgisayar ekranlarında 1 inç başına 96 nokta düşer. High-Res ekranlar ve printer gibi **yüksek çözünürlüğün** geçerli olduğu cihazlarda ise değer birden çok aygıt pikseline karşılık gelir.



**2) GÖRECELİ (RELATIVE) ÖLÇÜ BİRİMLERİ**

Göreceli CSS uzunluk birimleri sabit değerlere sahip değildir. Değerleri **önceden tanımlanmış**(predefined) başka bir değer veya özelliğe göredir. Göreceli birimler web tasarımında daha kullanışlıdır. Çünkü genişlik, yükseklik, yazı tipi boyutu vb. diğer bazı temel parametrelerle ilişkilendirebildiğimiz için öğeleri **düzgün şekilde** boyutlandırmayı kolaylaştırırlar.

**Yazı Tipi Göreli(Bağıl) Uzunluklar**

Yazı tipi göreli birimleri, önceden tanımlanmış yazı tipi boyutu veya yazı tipi ailesi değeri ile ilgilidir ve şunları içerir:

**em**

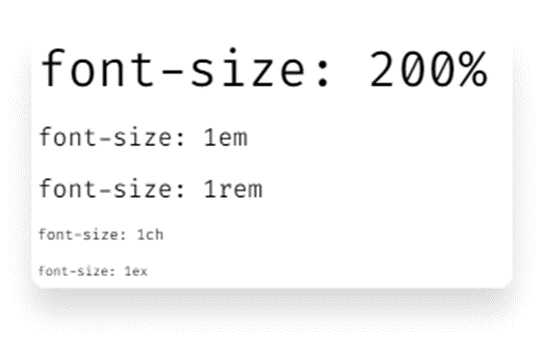
Bu ölçü biriminde varsayılan font ailesinde *(font-family)* yer alan *M***harfinin** **genişliği** ölçü için referans olarak alınır. **em** birimi, temel elementin (base element) veya üst elementin (parent element) yazı tipi boyutuna eşit bir değer alır. Örneğin, üst elementin yazı tipi boyutu 20px ise, 1em tüm alt elementler için 20px (20×1) olarak hesaplanır. Değerin bir tam sayı olmasına gerek yoktur. Yani 0,5em şeklinde de bir tanım yapmamız mümkündür. Bu durumda alacağımız birim karşılığı 10px olacaktır.

Unutulmaması gereken önemli nokta ise **em** değerinin **hemen üst etiketin**değerini almasıdır. Bu, iç içe geçmiş elemanlarda istenmeyen sonuçlara yol açabilir.

Örneğin iç içe geçmiş 3 elementimiz var. İlk eleman (root)20px font boyutunda, iç içe iki eleman ise fontu 2em olarak ayarlanmış. Kök öğenin içine yerleştirilen öğenin yazı boyutu 40 px (20 x 2) olurken iç öğenin içine yerleştirilmiş öğenin yazı boyutu 80 px (40 x 2) olur.

**rem (root em)**

**rem** birimi **em** biriminden farklı olarak her zaman kök elementin (HTML belgelerinde **html** öğesidir.) değerini alır ve art arda sıralı **em** gibi kullanılmaz. rem birimi özellikle **duyarlı tasarım** sürecinde oldukça faydalıdır. rem ile HTML elementinin yazı tipini değiştirerek tüm sayfanın ölçeklendirilmesi sağlanabilir.



**ex (x-height)**

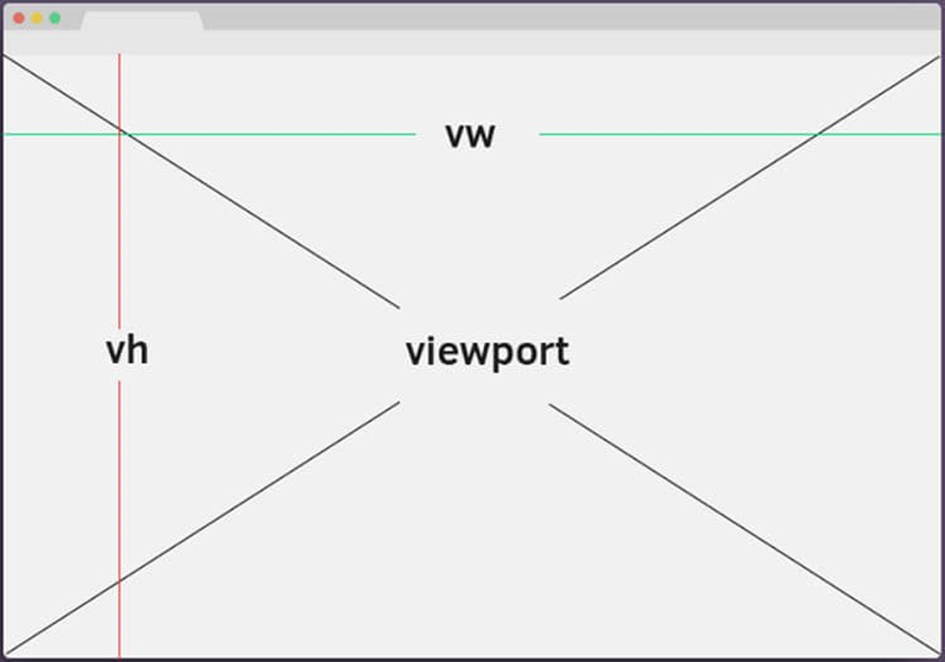
**ex**birimi **em** ile aynı mantıkta çalışır ,“geçerli yazı tipinin x yüksekliği” veya bir “**em**‘in yarısı” olarak tanımlanır. Belirli bir fontun x yüksekliği ise, o fontun küçük harf x’inin yüksekliğidir. Yazı tipi ailesi değiştikçe değişir. Nadiren kullanılan bir birimdir.

**ch (character)**

**ch** birimi ex birimine benzer ancak *küçük harf x*karakterinin yüksekliği yerine *0 (sıfır)* karakterinin **yüksekliğini** alır. Yine font ailesine bağlı olarak değişkenlik gösterir.

**Pencere Boyu veya Bakış Alanı Birimi (Viewport)**

Viewport, tarayıcı ekranın kullanıcı tarafından görünen kısmını ifade eder. HTML5, web tasarımcılarının *< meta >* etiketi ile görüntüleme alanı üzerinden kontrol sahibi olmamızı sağlayan bir yöntem geliştirmesi sayesinde bu yeni ölçü birimini CSS’e kazandırmış oldu.



Tasarımcılar için büyük kolaylık sağlayan bu ölçü birimi, artık javascript ile yapılan ekrana sığdırma gibi bazı işlemleri otomatik olarak yapmamıza olanak tanımaktadır.

İlk başta bakış alanını belirleyip sonra sayfa boyutunu değiştirdiğinizde buna göre yeni bir ölçeklendirme yapılır. Kullanıcı tarayıcı boyutunu her değiştiğinde hesaplamalar tekrar tekrar yapılır.

**Viewport olarak kullanabileceğiniz ölçü birimleri***vh, vw, vmin***ve***vmax***‘dir.**

**vw (viewport width)**

**vw**, görüntü alanı genişliği birimidir. 1vw görünüm alanının 1/100’üne eşittir. Örneğin pencerenin genişliği 1000px ise 1vw 10px’e karşılık olacaktır. **vw** temel olarak yüzde ifadesine benzese de üst elementlerden (parent) veya üst elementlerin genişliğinden bağımsız işler.

**vh (viewport height)**

**vh**, görüntü alanı yüksekliği birimidir ve **vw** ile benzer kurallarda işler, yani 1vh görünüm alanının 1/100’üne eşittir. Örneğin pencerenin yüksekliği 1000px ise 1vh 10px olacaktır.

**vmin (viewport minimum)**

**vmin** görüntü alanı yüksekliği (vh) ve görüntü alanı genişliği (vw ) değerlerine bakar ve minimum olan değer üzerinden 1/100 olarak oranlar. Örneğin, 1900x1080px bir ekranda min değer 1080px olacaktır. Dolayısıyla 1vmin 10.8px olarak değerlendirilir.

**vmax (viewport maximum)**

**vmax** tıpkı **vmin** gibi görüntü alanının yükseklik ve genişlik değerini inceler ancak bu sefer max olan değer üzerinden işlem gerçekleştirir. Yani, yukarıdaki örnek üzerinden ilerleyecek olursak 1900x1080px bir ekranda 1vmax karşılığı 19px olacaktır.

·       **viewport meta etiketinin kuralı şu şekildedir;**

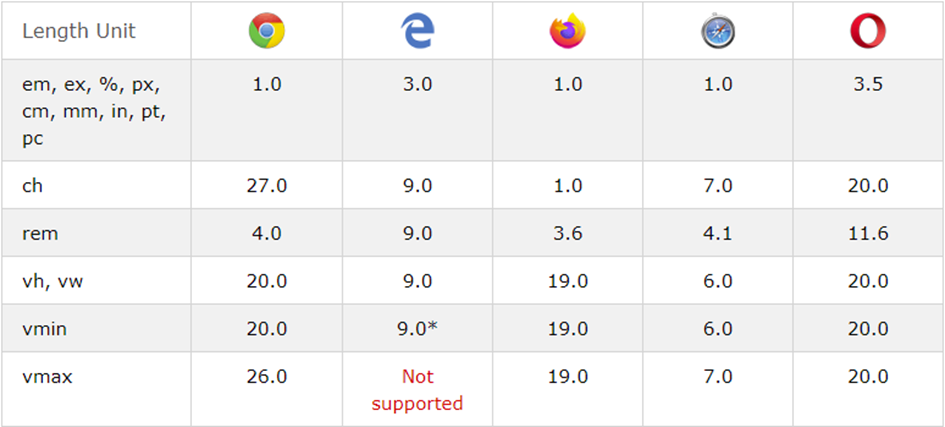
***<meta****name=”viewport” content=”width=değer, height=değer, initial-scale=değer, minimum-scale=değer, maximum-scale=değer, user-scalable=değer, target-desityDpi=değer”****/>***

**% (percentages)**

Başka bir uzunluğa bağlı olarak verilen yüzde orantı birimdir. Örneğin sayfamızda 500px genişliğinde bir alan olduğunu düşünürsek ve içerisine eşit boylarda 4 kutu oluşturacaksak kutuların genişliklerine “width: 25%;” değeri veririz. Böylece kutuların mutlak boyutunu 125px değerine eşitleyebiliriz. Yüzde ölçülendirme birimi **responsive tasarım** anlayışının en temel ölçü birimidir.

**Tarayıcı desteği**

Son olarak bu ölçü birimlerinin tarayıcı desteği ise şu şekilde;

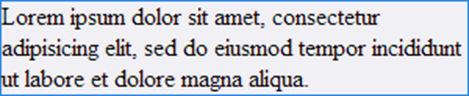


[Padding Nedir? Margin Nedir?](https://hakantasan.com/index/makaleler/14/css-padding-nedir-margin-nedir/)

Site yapımında en çok kullanılan ve birbiriyle çok karıştırılan 2 CSS özelliğinden bahsedelim; **padding** ve **margin.** Kısaca **padding**; element kenarı ile element içeriği arasındaki boşluktur. Kısaca **margin**; 2 elementin birbirine olan uzaklığıdır (boşluğudur). Şimdi örneklerle daha detaylı inceleyelim.

**1) Padding**

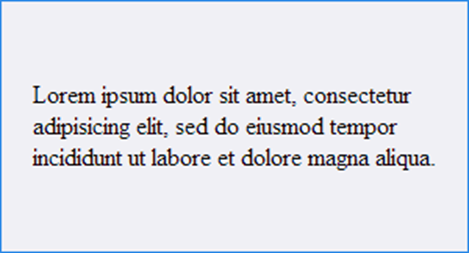
Element kenarı *(div, p, img vb. içeriği)* ile element içeriği *(div, p, img vb. içeriği)* arasındaki boşluktur (iç kenar boşluğu). Üst, alt, sağ ve soldan boşluk verilebilir. Aşağıdaki resimde arka planı gri, kenarlığı mavi renkte olan ve içinde anlamsız kelimeler yazan bir div var.



Padding'in çeşitli şekilde kullanımları vardır.

**a) 4 değerli padding**

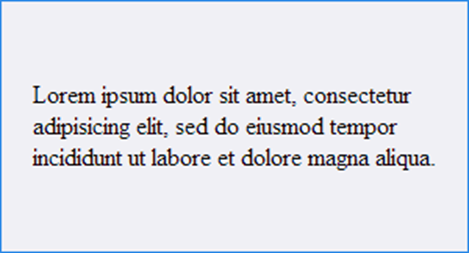
1.   padding:50px 20px 50px 20px;



Bu kod içeriğin yukarısından 50 piksel, sağından 20 piksel, aşağısından 50 piksel ve solundan 20 piksel boşluk bırakır. Daha kolay şekilde aklınızda tutabilmeniz için ilk değer yukarıdan başlayacak şekilde saat yönünde ilerlediğini düşünün. Değerler sadece piksel cinsinden olmak zorunda değildir, bunun dışında sık kullanılan Points (pt), Ems (em), Percent (%) vb. gibi cinsler vardır.

**b) 3 değerli padding**

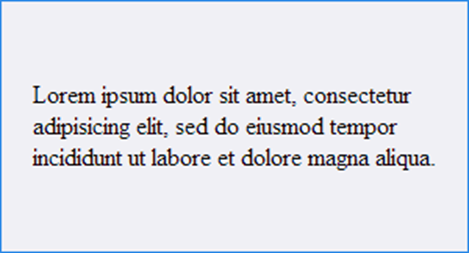
1.   padding:50px 20px 50px;



Bu kod içeriğin yukarısından 50 piksel, sağından ve solundan 20 piksel, aşağısından 50 piksel boşluk bırakır.

**c) 2 değerli padding**

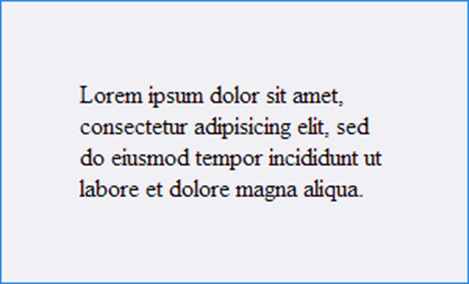
1.   padding:50px 20px;



Bu kod içeriğin yukarısından ve aşağısından 50 piksel, sağından ve solundan 20 piksel boşluk bırakır.

**d) 1 değerli padding**

1.   padding:50px;

****

Bu kod içeriğin her tarafından 50 piksel boşluk bırakır.

**e) Bağımsız padding kullanımı**

Bu şekilde bağımsız olarak padding değerleri ayarlanabilir.

**Yukarıdan boşluk bırakma**

1.   padding-top:100px;

**Sağdan boşluk bırakma**

1.   padding-right:100px;

**Aşağıdan boşluk bırakma**

1.   padding-down:100px;

**Soldan boşluk bırakma**

1.   padding-left:100px;

Örneğin; sadece soldan 20 piksellik boşluk bırakılacaksa **padding:0 0 0 20px;** yerine **padding-left:20px;** yazılabilir.

**2) Margin**

2 element içeriğinin (*div vb.*) birbirine olan uzaklığıdır (boşluğudur). Element kenarının dışında boşluk oluşturur. Üst, alt, sağ ve soldan boşluk verilebilir. Kullanımı şekli padding ile aynıdır.

1.   margin:100px 50px 100px 50px;

2.   margin:100px 50px 100px;

3.   margin:100px 50px;

4.   margin:100px;

5.

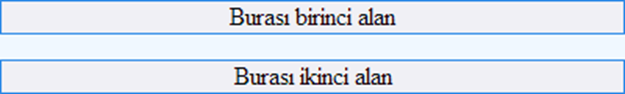
6.   margin-top:100px;

7.   margin-bottom:100px;

8.   margin-right:50px;

9.   margin-left:50px;

Aşağıdaki resimde arka planı gri, kenarlığı mavi renkte olan ve içinde yazılar yazan 2 div var.



**Örnek: margin:100px 50px 50px;**

Aşağıdaki resmi tanımlayacak olursak; ikinci div alanının üzerinde bir içerik olacaksa eğer ikinci alanın 100 piksel üzerinde olmalı, ikinci alanın sağında veya solunda bir içerik olacaksa 50 piksel sağında veya solunda olmalı, ikinci alanın altında bir içerik olacaksa 50 piksel altında olmalı.

