

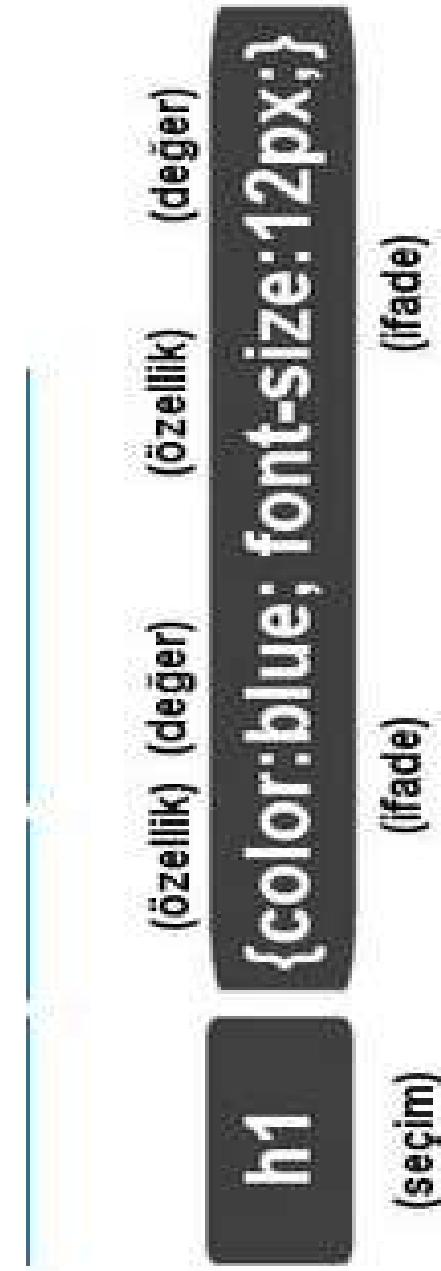
# CSS

- CSS bir kısaltmadır. "Cascading Style Sheets" kelimesinin baş harflerinden oluşur.
- Dilimizdeki anlamı: Basamaklı Stil Sayfası.
- Stiller, bir HTML elementinin nasıl görüneceğini belirleme olanağı sağlar.
- Stiller, bir HTML elementinin nasıl görüneceğini belirleme olanağı sağlar.

# CSS Kod Yapısı

- Stil dosyalarımızın uzantısı .css’ dir. CSS Dosyası birçok stil barındırabilir.
- Stiller css uzantılı dosyalara kaydedilerek kullanılır, görüntüyü hızlı ve etkili bir şekilde değiştirmenize yarar, düzenlemesi kolaydır, web sitenize kod fazlalığı yaratmaz.

- Bir CSS kodu iki temel bileşenden oluşur. Bir element ve stilleri birbirinden ayıran bir seçim. Diğer ise bir ya da daha fazla özelliği bildirdiğimiz ifade bölümü.

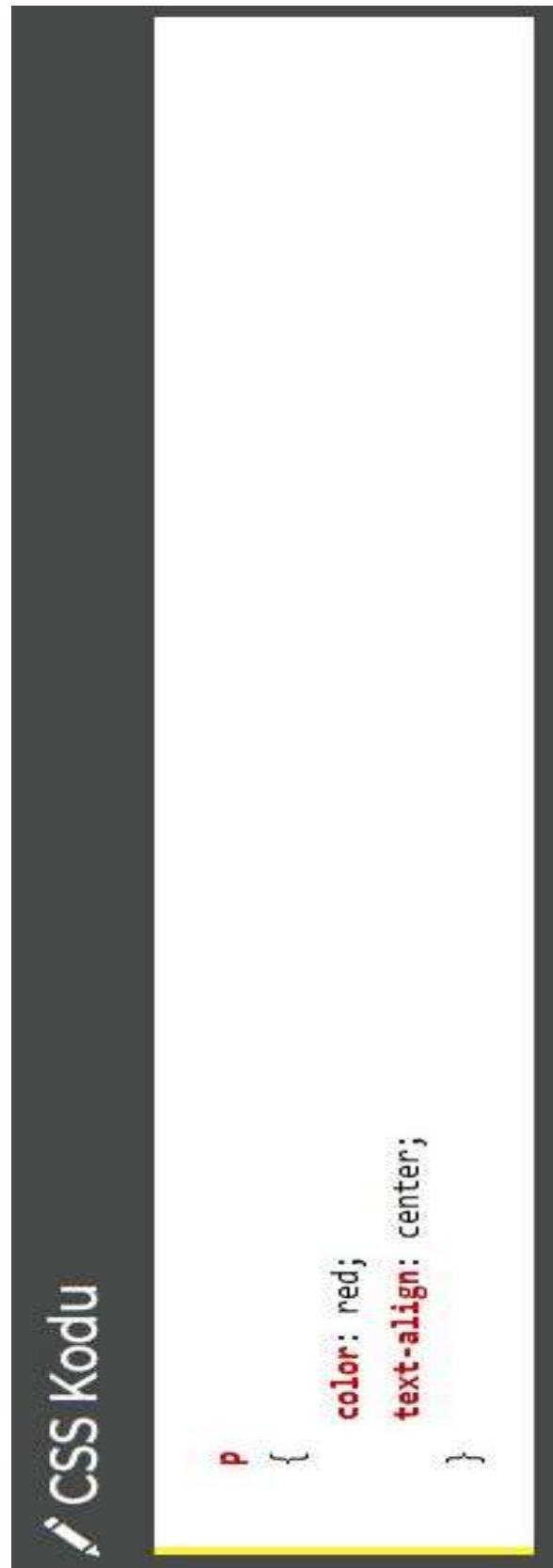


- Seçim genel olarak stilini belirleyeceğimiz HTML elementidir, h1, a, body, p gibi.
- İfadeler ise sürekli olarak **Özellik: değer;** şeklinde { ve } işaretleri arasında sıralanırlar.
- Bir css ifadesi her zaman noktalı virgül (;) ile biter. İfadeler bir kıvrımlı parantez içinde yer alır. Aşağıdaki örneğe bakalım:



```
p { color: red; text-align: center; }
```

- Bir önceki sayfada belirttiğimiz kodları istersek aşağıdaki gibi şekilde de yazabiliriz. Daha anlaşılır olacaktır:



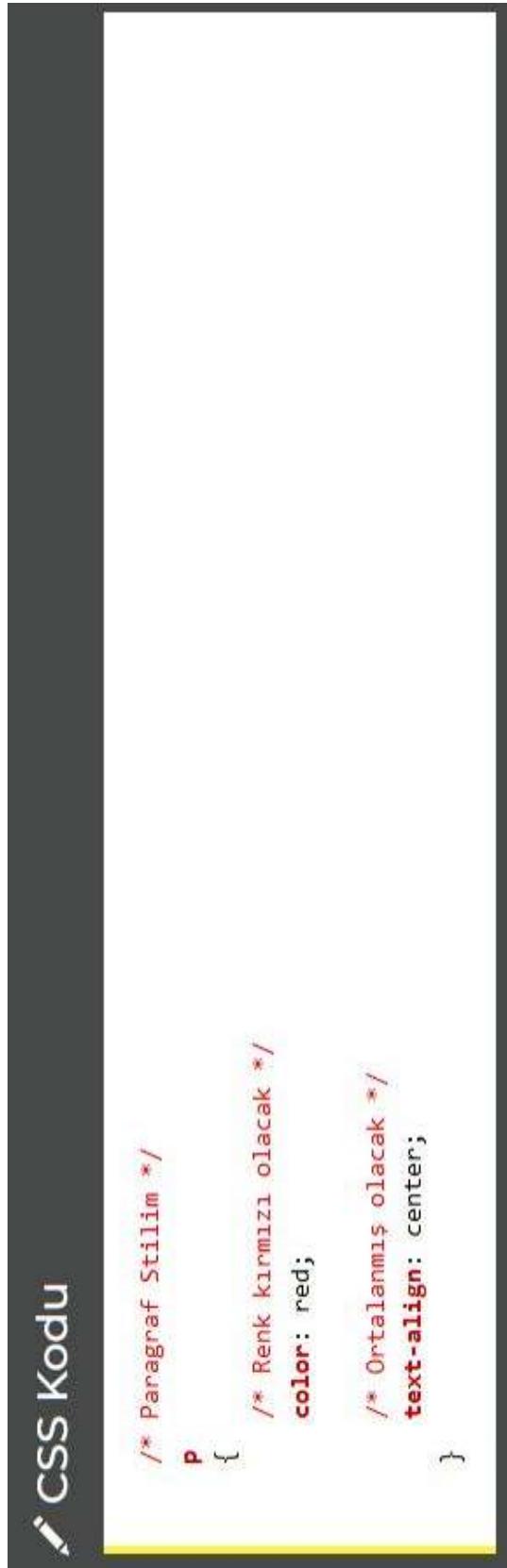
The screenshot shows a code editor interface with a dark theme. On the left, there is a sidebar with the text 'CSS Kodu'. The main area contains the following CSS code:

```
p {  
    color: red;  
    text-align: center;  
}
```

# CSS'de Açıklama(Yorum Satırı)

## Kullanmak

- Her programlama dilinde ve HTML'de de olduğu gibi bazen kullanıcıların görmeyeceğini ancak bizim için referans olacak açıklamalar yazmak gerekebilir. Bu durum için CSS de /\* açıklama \*/ kullanılmaktadır.



```
/* CSS Kodu */

/* Paragraf Stili */
p {
    /* Renk kırmızı olacak */
    color: red;

    /* Ortalanmış olacak */
    text-align: center;
}
```

# CSS ID ve Class Seçimi

- HTML elementleri için stiller belirleyebildiğiniz gibi size özel stiller belirleyip elementlerin "id" ve "class" özelliklerinde bu stili belirterek de kullanabilirsiniz.
- 1. Class Kullanımı
- Bir HTML kaynağında baktığınızda bir element (tag) Özelliği olarak class="stil" şeklinde bir özellik belirtildiğini görebilirsiniz. Peki neden buna gerek duymuşlar? HTML kodunu inceleyelim:

```
<div class="stil">Hoş Geldiniz!</div>
```

- Bir önceki sayfada div elementi içinde yer alan **class="stil"** , o div elementi için CSS Özellikleri belirtmemizi sağlayan bir yoldur. CSS dosyamızda şu şekilde bir ifade varsa:

```
.stil {  
    font: 10pt Tahoma, Verdana;  
    color: blue;  
}
```

Nokta (.) ile başlayan bir stil oluşturduk ve adını biz verdik. Bu demektir ki **class="stil"** ile belirttiğimiz tüm elementler 10 punto Tahoma yazı tipinde ve mavi (blue) renginde olacaktır. Tabi istersek sadece tek bir elementte geçerli olmasını sağlayabiliriz. Örneğin sadece DIV elementlerinde geçerli olmasını istiyorsak:

```
div.stil {  
    font: 10pt Tahoma, Verdana;  
    color: blue;  
}
```

# CLASS Özelliği kullanımının bize sağlayacağı yararlar

- ".stil" yani seçim adımızın başına div getirmemiz yeterli olacaktır. Bu durumda bu CSS kodu sadece DIV elementlerinde kullanılabilir olacaktır.
- Kendimizin adlandırdığı özel stiller yaratmak ve kullanmak
- Bir stili birden fazla elementte kullanabilmek
- Stillere CSS de yer verip HTML kodlarımızı sürekli tekrarlanan uzun CSS kodlarından arındırmak.

# ID Kullanımı

- ID Özelliği ile de stiller yaratılabilir. CLASS'tan farklı yanları:
- Sadece tek bir elementte kullanılabilir.
- Aynı id değeri iki elemente verilemez (Her id sadece tek bir elementte kullanılabilir).
- Stil dosyamızda Class'da . (nokta) kullanırdık, ancak id özelligine göre stilleme yapacaksak # (diyez) kullanırız.

# Şimdi bu anlatıklarımızı örnek üzerinde görelim:

```
<div id="stil">Hoş Geldiniz!</div>
```

- Örnekte ID değeri "stil" olan bir DIV elementi görüyoruz. CSS dosyamızda bu elemente özel stil tasarlarken aşağıdaki gibi şekilde kod yapısını oluşturmalıyız:

```
#stil {  
    font: 10pt Tahomā, Verdana;  
    color: blue;  
}
```

- Görüldüğü gibi bu kez diyez (#) ile başlattık.
- Sadece tek bir elemente özel stil tasarlamış olduk (Neden? Çünkü başka bir elementte yine id değeri "stil" belirtilemez.).
- Ek olarak, hiçbir ID değeri rakamla başlamaz. ID, HTML dosyasında o elementi bulmamızı sağlar, bu nedenle aynı ismi veremeyiz. Aynı ismi vermemizin diğer yaratacağı sorun Java Script'te bu elementi id özelliğine göre kullanamak olacaktır. Son olarak bazı tarayıcılar birden fazla aynı id kullanılan HTML dosyalarında stilleri göremezden gelebilir.

# CSS Nasıl Eklenir ?

- Bir tarayıcı açıldığı zaman stilleri okur ve o stilin kullanıldığı HTML elementlerini belirtilen özelliklere göre Şekillendirir.
- Üç şekilde sayfamıza stil ekleyebiliriz. Bunlar:
  - Stilleri CSS Dosyasından ÇağırmaK
  - HTML Sayfasında CSS Yazmak
  - HTML Elementinin İçerisinde Stil Belirtmek

# 1. Stilleri CSS Dosyasından Çağırma(En kullanımı yol)

- Öncelikle bir not defteri ya da CSS düzenleyici program açmalısınız.
- Visual Studio Code programında yeni dosya oluştur kismından dosyanızı oluşturun ve ismini stil\_dosyasi.css yapın.

CSS Kodlarınızı CSS kod yapısına uygun olarak yazın ve kaydedin.

HTML Sayfanızı açıp <head> ile </head> arasına aşağıdaki Sekilde stil dosyanızın adını belirtin.

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

- href="DOSYA ADI" ile belirttiğimiz yerde bizim kaydettiğimiz dosyanın adı yer almalı ve HTML sayfası ile CSS dosyası aynı klasörde bulunmalıdır. Farklı bir klasörde kullanmak istiyorsak HTML sayfasına göre konumu yazmamız gereklidir.
- Dosya adını doğru yazdiysak artık CSS dosyamızı HTML sayfamıza bağlılık demektir. Yani CSS dosyamızdaki stilleri rahatlıkla HTML sayfamızda kullanabiliriz.

## 2. HTML Sayfasında CSS Yazmak

- HTML sayfamızda HEAD elementlerinin arasında STYLE elementi kullanarak stiller yaratmamız mümkün. Aşağıdaki örneğe bakalım:

### HTML Kodu

```
<html>
<head>
<title>Sayfa Başlığı</title>
<style type="text/css">
body { background-color:black; color:white; }
p { font-family: Tahoma, Verdana; font-size: 12px; }
</style>
</head>
<body>
<p>Bu 12 piksel Tahoma yazı tipi ile yazıldı.</p>
</body>
</html>
```

# 3. HTML Elementinin İçerisinde Stil Belirtmek

- Bazen stil dosyası ya da STYLE elementi kullanmadan hızlı çözümler üretmek gerekebilir. Böyle durumlarda her elementin style="" özelliği kullanıma hazırlıdır. CSS kodlarını element içinde açacağımız STYLE özelliğine sıralarız. Örneğin;

İ HTML Kodu

```
<p style="font-family: Tahoma; font-size: 12px;">  
Bu 12 piksel Tahoma yazı tipi ile yazıldı.  
</p>
```

# CSS Arkaplanlar

- CSS arkaplan özelliklerini bir HTML nesnesine arkaplan eklemenizde yardımcı olur. Bununla ilgili olarak tanıyalım CSS kodları:
  - background-color
  - background-image
  - background-repeat
  - background-attachment
  - background-position
  - background

# 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: #00000000 şeklinde önce diyez sonra 6 adet 0-9 ve A-F değerleri alabilen sembollerden oluşur. Örneğin #404040 bir renki ifade etmektedir.

- RGB (Kırmızı, Yeşil, Mavi) Renkler: Bir renk oluştururken kırmızı, yeşil ve mavinin tonlarını belirterek renk oluşturmamızı sağlar. Örneğin `rgb(0,0,0)` siyah renki belirtir. Mavi bir renk elde etmek istediğimizde `Red - Green - Blue` üclemesinde en sonda yer alan mavinin değerini arttırmamız yeterli olur. Yani `rgb(0,0,255)` mavi renki verir. Windows'daki Paint Brush (MS Paint) programı ve diğer resim editörleri yardımıyla RGB değerlerini alarak renk oluşturabilirsiniz.

- Örnek RGB Renk Hesaplayıcısı:

- [https://www.w3schools.com/colors/colors\\_rg\\_b.asp](https://www.w3schools.com/colors/colors_rg_b.asp)

- Tarayıcı Renk İsimleri: İngilizce olarak belirlenmiş bazı renk adlarını yazarak renk elde edebilirsiniz. Örneğin white beyaz, black siyah, blue mavi rengini verecektir.

# Örnek Renk Kullanımı

## İÇSS Kodu

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

/* Yeşilin bir tonu */
div { color: lime; }

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

## **background-color: Arkaplan Renği**

- Bir elementin arkaplan rengini belirlememizi sağlar.



The image shows a screenshot of a code editor. On the left, there is a dark sidebar with the text 'CSS Kodu'. The main area is white and contains the following CSS code:

```
div { background-color: #EFEFEF; }
```

# background-image: Arkaplan Resmi

- Bir element içinde arkaplan resmi kullanmamızı sağlar. Aşağıdaki gibi url("") yazılarak arasında resmin adı veya yolu yazılır. Örnekte resmin adı resim.jpg olarak gösterilmiştir.

i CSS Kodu

```
body { background-image: url('resim.jpg'); }
```

# **background-repeat: Arkaplan Tekrarı / Döşeli**

- Kullandığımız arkaplan resminin tekrar edip etmeyeceğini belirtmemizi sağlar. Dört kullanımı vardır:
- no-repeat : Tekrar edilmeyeciktir
- repeat : Tekraranacaktır / döşenecektir
- repeat-x : Sadece sağa doğru tekrar edecektir
- repeat-y : Sadece aşağı doğru tekrar edecektir

Aşağıdaki kullanım örneğini inceleyelim: Örnek CSS kodu resmin tekrarlanmayağını tarayıcıya bildirir.

i CSS Kodu

```
body {  
background-image: url('resim.jpg');  
background-repeat: no-repeat;  
}
```

# **background-attachment: Arkaplan Sabitliği**

- Bu kod kullandığımız arkaplanın sabit kalıp kalmayacağı hakkında ayar yapmamızı sağlar. Eğer özellik değerinin sabit kalmasını (yani sayfanın yerine göre değişmemesini) istiyorsak fixed özelliği kullanırız.

 CSS Kodu

```
body {  
    background-image: url('resim.jpg');  
    background-attachment: fixed;  
}
```

# **background-position:** Resmin Nereye Hizalanacağı

- Eğer background-repeat özelliğini no-repeat olarak belirlediysek bu kod yardımıyla onun hizalanma şeklini belirleme şansına sahip oluruz.
- Aşağıda yer alan kod arkaplan resminin right (sağ) ve top (üst) tarafa doğru hizalanacağını bildirir.

i CSS Kodu

```
body {  
  
background-image: url('resim.jpg');  
  
background-repeat: no-repeat;  
  
background-position: right top;  
  
}
```

# background: Arkaplan Belirlemenin Kısa Yolu

- Yukarıdaki tüm özellikleri tek bir kodda kullanmanız mümkün. Bunun için kullanacağımız kod: background.
- Aşağıdaki kodda belirtilenler sırasıyla: background-color, background-image, background-repeat ve background-position.

css Kodu

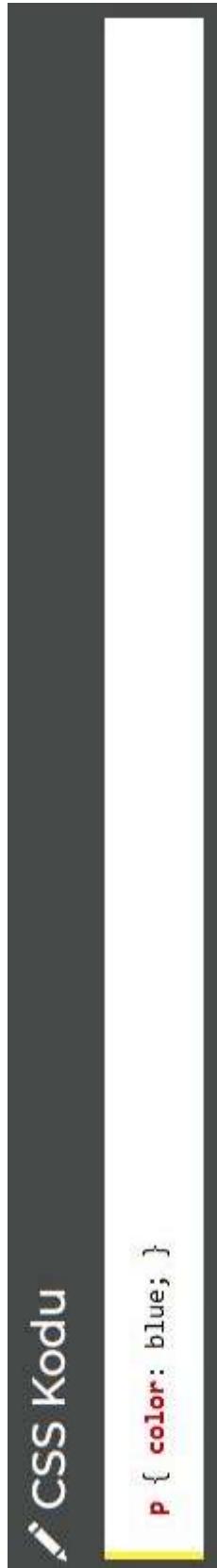
```
body {  
background: #000000 url('resim.jpg') no-repeat right top;  
}
```

# CSS Yazılar

- Bu sayfada HTML sayfanızda kullandığınız yazıların değiştireceğiniz belki başlı özellikleri (renk, hiza, kalın vb) hakkında bilgi verecektir.
- Yazı biçimlendirme ile ilgili komutlardan tanıyacaklarımız:
  - color
  - text-align
  - text-decoration
  - text-transform
  - text-indent

## color: Yazı Renği

- Yazının rengini belirlemenizi sağlar.



```
! CSS Kodu
p { color: blue; }
```

## text-align: Hizalama

Yazının yatay yönde ne şekilde hizalandacağını belirtmenizi sağlar. En çok kullanılan dört hizalma yöntemi:

left : sol

right : sağ

center : ortalanmış

justify : iki yana yasla

# text-decoration: Yazı Biçimi

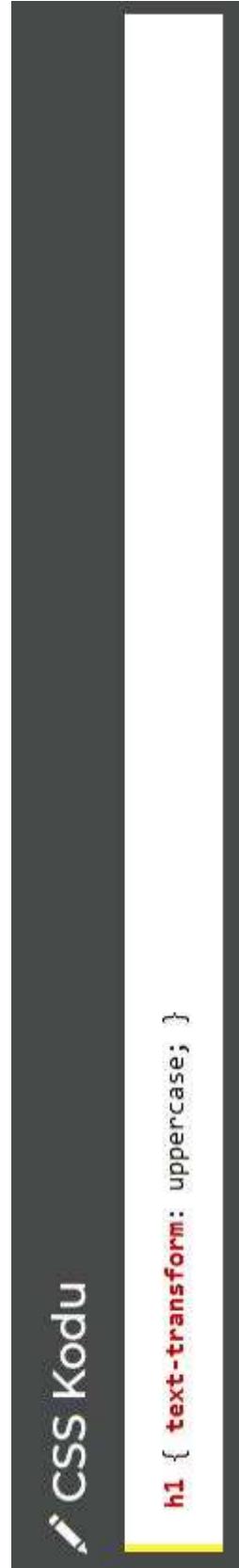
- Yazının biçiminde bazı değişiklikler yapmamızı sağlar. Örneğin tarayıcı varsayılanında sürekli A elementlerindeki altı çizgili dekorasyonu kaldırmak için aşağıdaki gibi yazılabılır.

The screenshot shows a dark-themed code editor window. On the left, there is a sidebar with a pencil icon and the text 'CSS Kodu'. The main area contains the following CSS code:

```
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



The screenshot shows a dark-themed code editor interface. On the left, there is a sidebar with the text "CSS Kodu". The main area contains the following CSS code:

```
h1 { text-transform: uppercase; }
```

- 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 boşluk koyar.
- word-spacing: Kelimeler arasında belirtilen boşluk koyar.

# CSS Yazı Tippleri

- Bu bölümde HTML sayfanızda kullandığınız yazıların yazı tipi özelliklerini nasıl değiştireceğiniz hakkında bilgi verilecektir.
- Yazı tiplerini biçimlendirirken en sık kullandığımız komutlar:
  - font-family
  - font-size
  - font-style
  - font-weight

# font-famililiy: Yazı Tipi Belirlemek

- Yazı tipleri klasörümüzü açtığımızda birçok yazı tipi görürüz. Bunları HTML sayfamızda kullanmak için adını belirtmemiz gereklidir. İşte bu kod yazı tipi seçmemizde bize yardımcı olacaktır. Sayfamızda kullanmamızı istediğimiz yazı tiplerimizi bulabileceğimiz en Önemli kaynak google fonts ‘dur.
- <https://fonts.googleapis.com/knowledge>

```
h1 { font-family: Times New Roman; }

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

a { font-family: Sans-serif; }
```

- Yukarıdaki birinci örnekte sadece tek yazı tipi adı belirttiğimiz (Times New Roman), ancak bir sonrakinde aralarına virgül koyarak daha fazla belirttiğimiz yazının öncelikle dikkate alınan ilk yazının yazısı tipi olacaktır. Eğer verdiğimiz yazı tipi o kullanıcida yoksa bir sonraki ile görüntülenecektir.

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

- Yazı büyülüüğünü font-size ile belirtiriz. Ölçü birimi olarak aşağıdakileri kullanabiliriz:
  - % : yüzde olarak belirtmek
  - pt : punto olarak belirtmek
  - px : piksel olarak belirtme
- Örneğin yazımızın 12 punto olması için yazmamız gereken komut:

 CSS Kodu

```
h2 { font-size: 12pt; }
```

- 14 piksellik bir yükseklikle sahip yazı istiyorsak:

A screenshot of a code editor interface. On the left, there is a dark sidebar with a white 'CSS Kodu' button. The main area is a light-colored text editor. Inside, there is a single line of CSS code: `h2 { font-size: 14px; }`. The entire screenshot is framed by a thick black border.

```
h2 { font-size: 14px; }
```

Yukarıdaki örneği yüzde ile ifade etmek isteseydik:

A screenshot of a code editor interface. On the left, there is a dark sidebar with a white 'CSS Kodu' button. The main area is a light-colored text editor. Inside, there is a single line of CSS code: `h2 { font-size: 150%; }`. The entire screenshot is framed by a thick black border.

```
h2 { font-size: 150%; }
```

- **font-style: italic (Sağa Yatık) Yazmak**
  - Yazımızın sağa yatık (italic) olması için aşağıdaki kodu kullanabiliriz.

✓ CSS Kodu

```
h2 { font-style: italic; }
```

- Eğer italic bir yazıyı iptal edeceksek aşağıdaki kodu yazmalıyız:

✓ CSS Kodu

```
h2 { font-style: normal; }
```

# font-weight: Kalın Yazmak

- Yazımızın kalın (**bold**) olmasını istiyorsak aşağıdaki kodu kullanmalıyız.

 CSS Kodu

```
h2 { font-weight: bold; }
```

- Yine aynı şekilde eğer kalın bir yazıyı normal haline dönüştüreceksek aşağıdaki kodu yazmalıyız:

 CSS Kodu

```
h2 { font-weight: normal; }
```

# CSS'de <a> Etiketi

- Web sayfaları arasında geçişyi sağlamak yani link vermek için <a> etiket kullanılır. <a> etiketi içerisinde metin, resim ya da diğer HTML etiketleri bulunabilir.

Yapısı (Syntax)

```
<a href="">...</a>
```

HTML

- Eğer href belirtmemiş ise, download, hreflang, media, rel, target, ve type nitelikleri işe yaramaz.
  - \* Mevcut bağlantıya tıklandığında aksi belirtmediği takdirde (rel) standart olarak mevcut pencere içerisinde açılır.

# A etiketi için varsayılan CSS değerleri

```
css
a {
  cursor: pointer;
  text-decoration: underline;
}
```

- En basit haliyle kullanımı;

```
html
<p> <a href="https://prototurk.com">Prototürk</a>, web dilleriyle ilgili türkçe kaynaklar üretir.
</p>
```

- Bağlantı içerisinde resimli kullanımı;

```
html
<a href="https://prototurk.com">
  
</a>
```

# CSS Listeleme

- Listelerde kullandığımız madde imlerini biçimlendirirken CSS'den faydalananabiliriz.
- CSS üç bakımından işimize yarar:
  - Sıralı listelerdeki numaraların görünüşünü değiştirebiliriz.
  - Maddelenmiş listelerdeki imlerin Şekillerini değiştirebiliriz.
  - Madde imi yerine belirlediğimiz bir resim kullanabiliriz.

# Listelerde biçimlendirme yaparken Sıklıkla kullanılan komutlar:

- list-style-type
- list-style-image
- list-style-position
- list-style (Birleştirilmiş)
- **list-style-type: Listeleme Şekli Tipi**
- Bir listenin stilini belirlememizi sağlar. Aşağıdaki örnekte madde işaretleri kare (square) olacaktır:

» CSS Kodu

```
ul { list-style-type: square; }
```

# HTML GÖRÜNÜMÜ AŞAĞIDAKİ ŞEKİLDE OLACAKTIR.

- Maddelenmiş Liste
- Numaralanmış Liste

- Bir web sitesinde bir liste kullanmak istediğimizde HTML, seçim için üç farklı tür sunar;
- Sırasız (Unordered List-ul)
- Sıralı (Ordered List-ol)
- Açıklama listeleri (Description List-dl)

- Kullanılacak listenin türünü seçmek ya da bir liste kullanmak isteyip istemediğinizi seçmek, içeriğe ve içeriği görüntülemek için yapacağınız tercihlere bağlıdır.
- HTML'de bulunan üç farklı türde listeyi, CSS ile stillemek için birçok yol bulunmaktadır. Örneğin, bir listede kullanılacak madde imi türünü seçebiliriz. İşaretçi, kare, yuvarlak, sayısal, alfabetik veya belki de görünmez olabilir.

- Not : <ol> veya <ul> etiketine uygulanan herhangi CSS bildirimini tüm listeyi etkilerken, <li> etiketine uygulanan özellikler tek tek liste öğelerini etkiler.
- list-style-type: list-style-type Özelliği liste öğesinin başında yer alan işaretçiyi (item marker) değiştirmenizi sağlar. Alabileceğiniz CSS değerleri;

none	Liste öğesi yok
disc	Dolu bir daire
circle	İçi boş bir daire
square	Dolu bir yuvarlak

- Aşağıdaki Örnekte yer alan liste öğelerine değişik liste stil tipleri uygulanmıştır.

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <style>
5  ul.bir {
6      list-style-type: circle;
7  }
8
9  ul.iki {
10     list-style-type: square;
11 }
12
13 ol.uc {
14     list-style-type: square;
15 }
16
17 ol.dort {
18     list-style-type: upper-roman;
19 }
20
21 </style>
22 </head>
23 <body>
24
25 <p>Sırasız Listeler</p>
26 <ul class="bir">
27 <li>Çay</li>
28 <li>Kahve</li>
29 <li>Soda</li>
30 </ul>
31
32 <ul class="iki">
33 <li>Çay</li>
34 <li>Kahve</li>
35 <li>Soda</li>
36 </ul>
```

# Web sitesimiz üzerinde görünümleri

## Sırasız Listeler

- Cay
  - Kahve
  - Soda
- Cay
  - Kahve
  - Soda

# list-style-type : none

- list-style-type Özelliğine none değerinin verilmesi <li> öğelerinizi özelleştirmenin yolunu açar. Madde imlerini kaldırır.

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  </head>
5  <style>
6  ul.ozel {
7      list-style-type: none;
8      margin: 0;
9      padding: 0;
10 }
11 </style>
12 </head>
13 <body>
14 <p>Normal Liste</p>
15 <ul>
16 <li>Çay</li>
17 <li>Kahve</li>
18 <li>Soda</li>
19 </ul>
20
21 <p>Özel Liste</p>
22 <ul class="ozel">
23 <li>Çay</li>
24 <li>Kahve</li>
25 <li>Soda</li>
26 </ul>
27
28 </body>
29 </html>
```

# Webstemizde Görünümleri

## Normal Liste

- Çay
- Kahve
- Soda

## Özel Liste

- Çay
- Kahve
- Soda

# CSS Kenarlıklar (Border)

- Bir HTML elementinin kenarlarına çizgi çekerken kullanacağımız kodlar şunlardır:
- border-style
- border-width
- border-color

# **border-style: Kenarlık Stiller**

- Kenarların görünüşünün nasıl olacağı ile ilgili bilgi vermemizi sağlar.
- Kullanabileceğiniz stiller:
  - none - Kenarlık yok
  - dotted - Noktalı
  - dashed - Kesik çizgili
  - solid - Çizgi
  - double - Çift çizgi
  - groove
  - ridge
  - inset
  - outset

# border-width: Kenarlık Boyutu

- Kenar genişliğini belirtmemizi sağlar.  
ÖNCELİKLE MUTLAKA border-style belirtmemiz gereklidir, aksi takdirde görüntülenmeyecektir.

✓ CSS Kodu

```
.kutum { border-style: solid; border-width: 1px; }
```

# border-color: Kenarlık Renkİ

- Kenarlık rengini bu komutla değiştirebiliriz.  
ÖNCELİKLE MUTLAKA border-style belirtmemiz gereklidir, aksi takdirde görüntülenmeyecektir.
- 

✓ CSS Kodu

```
.kutum {  
border-style: solid;  
border-width: 1px;  
border-color: #0000CC;  
}
```

# Kenarları Ayrı Ayrı Şekillendirmek

- İstiyorsak yukarı (top), aşağı (bottom), sağ (right) ve sol (left) kenarları ayrı ayrı biçimlendirebiliriz.
- border-left-style - Sol kenarın şekli.
- border-right-style - Sağ kenarın şekli.
- border-top-style - Üst kenarın şekli.
- border-bottom-style - Alt kenarın şekli.

`border-style` yazdıktan sonra sırasıyla saat yönünde kenarları ayrı ayrı belirleyebiliriz.

- Ayrıca `border-style` kenarın değerini de alacak şekilde yazılabılır. Örneğin:
  - Üst kenar - dotted,  
Sağ kenar - solid,  
Alt kenar - double,  
Sol kenar – dashed
  - **`border-style: dotted solid double dashed;`**

# Kenar Belirtmenin Kısa Yolu

- Sadece border kullanarak da border-width, border-color ve border-style değerlerlerini belirtmemiz mümkün. Örneğe baktırm:

✓ CSS Kodu

```
.kutum {  
    border: 1px #0000CC solid;  
}
```

# CSS Dıştan Boşluk (Margin)

- margin özelliği bir HTML elementinin çevresi ile ona komşu başka bir HTML elementi arasında belirtilen miktarда genişletip boşluk yaratır. Bu boşluk kenarlığın dışında olacağı için dış kenarlık diyoruz.
- Dış kenarlığı belirlerken üst, sol, alt ve sağ için ayrı ayrı belirleneceği gibi tek bir CSS koduyla da bunlar hızlıca belirlenebilir.

# Tüm margin komutları:

- margin - Hızlı kullanım
- margin-top - Üstten boşluk
- margin-right - Sağdan boşluk
- margin-bottom - Altan boşluk
- margin-left - Soldan boşluk

## Alabileceğimiz değerler:

- auto - Kenarlıklar tarayıcı tarafından otomatik ayarlanır
- px - piksel olarak boşluk belirtme
- % - İçerik genişliğine göre oranlar

# Ayrı Ayrı Dış Boşlukları Ayarlamak

- Üst, sağ, alt ve sol boşlukları ayrı ayrı ayarlanabilir. Bunun için margin komutuna araya tire (-) koyarak ilgili yönün ingilizcesini yazıyoruz.

/ CSS Kodu

```
.stilim {  
    margin-top: 10px;  
    margin-bottom: 10px;  
  
    margin-left: 20px;  
    margin-right: 20px;  
}
```

# Diş Boşlukları Ayarlamanın Kısa Yolu

- Sadece margin kullanarak hepsi için geçerli tek bir tane değer yazabildiğimiz gibi sırasıyla üst, sağ, alt, sol şeklinde değerler girerek de yukarıdaki dört kodun tamamını tek kodda yazabiliriz:

İ CSS Kodu

```
.stillim {  
    /* Açıklama: margin: [üst] [sağ] [alt] [sol] */  
  
    margin: 10px 20px 10px 20px;  
}
```

Yukarı ve alt değeri ile Sağ ve sol değeri aynı olan ifadelerde Şu Şekilde de kullanabileceğimizi unutmayalım

İCSS Kodu

```
.stillim
{
    /* Açıklama: margin: [üst|alt] [sağ|sol] */
    margin: 10px 20px;
}
```

# CSS İçten Boşluk (Padding)

- padding Özelliği bir HTML elementinin kenarlarının içinde bir boşluk yaratmamızı sağlar.
- Kenardan içerisindeki bu boşlukları belirlerken üst, sol, alt ve sağ için ayrı ayrı belirlenebileceği gibi tek bir css koduyla da bunlar hızlıca belirlenebilir.

# Tüm padding komutları:

- padding - Hızlı kullanım
- padding-top - Üstten boşluk
- padding-right - Sağдан boşluk
- padding-bottom - Altтан boşluk
- padding-left - Soldan boşluk

## Padding Alabileceği değerler:

- genişlik belirtme - px, em, pt gibi ölçülerle genişliği belirtme
- yüzde kullanma (%) - içerik genişliğine göre oranlar

# Ayrı Ayrı İç Boşlukları Ayarlamak

- Üst, sağ, alt ve sol boşlukları ayrı ayrı ayarlanabilir. Bunun için padding komutuna araya tire (-) koyarak ilgili yönün ingilizcесini yazıyoruz.

/ CSS Kodu

```
.stilim {  
    padding-top: 10px;  
    padding-bottom: 10px;  
  
    padding-left: 20px;  
    padding-right: 20px;  
}
```

# İç Boşlukları Ayarlamanın Kısa Yolu

- Sadece padding kullanarak hepsi için geçerli tek bir tane değer yazabildiğimiz gibi sırasıyla üst, sağ, alt, sol şeklinde değerler girerek de yukarıdaki dört kodun tamamını tek kodda yazabiliriz:

✍ CSS Kodu

```
.stilim
{
    /* Açıklama: padding: [üst] [sağ] [alt] [sol] */
    padding: 10px 20px 10px 20px;
}
```

- Yukarı ve alt değeri ile Sağ ve sol değeri aynı olan ifadelerde şu Şekilde de kullanabileceğimizi unutmayalım:

i CSS Kodu

```
.stillim
{
    /* Açıklama: padding: [üst|alt] [sağ|sol] */
    padding: 10px 20px;
}
```

# Gruplama

- Birden fazla stilde sıkılıkla kullanılan CSS komutları varsa, bunları toplu olarak yazıp özelliği bir kez yazma şansına sahibiz. Buna gruplama diyoruz. Örneğe bakalım:

!/ CSS Kodu

```
h1 { color: green; }
h2 { color: green; }
p { color: green; }
```

- Bir önceki sayfada görüldüğü gibi üç element için de tek bir özellik belirttilmiş. Bu da rengin yeşil olması. Bu kodu kısaltmak için element ve stil adlarını aralara virgül gelecek şekilde sıralayabiliriz. Şöyle ki:



css Kodu

```
h1, h2, p { color: green; }
```

# Üzerine Yazma

- Bazen tüm elementleri kapsayacak kodlar yazarız. Ancak bir ya da birkaç element bundan farklı olmalıdır. İşte bu durumda tüm elementlerde geçerli olan özelliklerini değiştirmemiz gerekebilir.

## HTML Kodu

```
<p>Bu bir paragraf</p>  
  
<p>Bu da bir paragraf ancak farklı bir stil olacak</p>  
  
<p>Bu da bir paragraf ve 1. paragraf ile aynı olacak</p>
```

- Tüm P elementleri için genel özelliklerini belirledikten sonra, farklı olmasını istediğimiz 2. paragraf özelliklerini yeni bir stil adı oluşturarak yazıyoruz. Geriye kalan ilgili paragrafa class="kirmizi" eklemek olacaktır.



The screenshot shows a code editor with a dark theme. On the left, there is a sidebar with the text "CSS Kodu". The main area contains the following CSS code:

```
P {  
    font: 10pt Tahoma;  
    color: gray;  
}  
  
.kirmizi P {  
    color: red;  
}
```

- Bir önceki sayfadaki kısım incelendiğinde tüm <P> elementleri 10 punto Tahoma ve gri olacaktır. Ancak class belirtip kırmızı yazdığımız P diğerlerinden farklı olarak kırmızı renkli olacaktır.

#### HTML Kodu

```
<p>Genel stile sahip paragraf</p>  
<p class="kirmizi">Kırmızı renkli olacak paragraf</p>  
<p>Diğer bir genel stile sahip paragraf</p>
```

# Genişlik ve Yükseklik

- Bir nesnenin genişliğini width, yüksekliğini height kodu kullanarak belirleriz. Alacağı değer piksel, punto, Yüzde gibi bir ölçü değeri olacaktır.
- Bu stili kullandığımız element 400x400'lük bir alanda yer alacaktır.

 CSS Kodu

```
.kutu {  
    width: 400px;  
    height: 400px;  
}
```

# En Düşük Genişlik ve Yükseklik Değeri

- Bir nesnenin minimum yani en düşük sahip olmak zorunda olduğu genişlik için min-width, yükseklik için min-height kodlarından faydalanırız.

 CSS Kodu

```
.kutu {  
    min-width: 100px;  
    min-height: 100px;  
}
```

# En Yüksek Genişlik ve Yükseklik Değeri

- Bir nesnenin maksimum yani en fazla sahip olabileceği genişlik için max-width, yükseklik için max-height kodlarından faydalanzı.

i CSS Kodu

```
.kutu {  
    max-width: 500px;  
    max-height: 200px;  
}
```

# Bir Elementin Genişlik ve Yüksekliği

- İçerik alanının genişliği width ve yüksekliği height özelliği ile ayarlanır.

i CSS Kodu

```
.kutum {  
width:250px;  
padding:10px;  
border:5px solid gray;  
margin:10px;  
}
```

- Bir önceki Örnekte genişlik (width) 250 piksel olarak ayarlanmıştır. Peki HTML dosyamızda da bu elementin kaplayacağı alan 250 piksel mi olacaktır?
- Hesaplama:
  - $250 \text{ piksel} = \text{Genişlik}$
  - $20 \text{ piksel} = \text{Sağdan ve Soldan İç Boşluk (padding)} \\ 10x2)$
  - $10 \text{ piksel} = \text{Sağ ve Sol Kenarlıklar (border)} 5x2)$
  - $20 \text{ piksel} = \text{Sağ ve Sol Dış Boşluk (margin)} 10x2)$
  - $300 \text{ piksel} = \text{Toplam Genişlik}$

# CSS Gösterme - Gizleme

- Bazen HTML elementlerini kullanıcıcıdan gizlemek gerekebilir. Bir nesnenin görünme şekli ve gizlenmesini display komutu kullanarak yaparız.
- **Bir Elementi Gizlemek/Göstermemek**
- Bunun için kullanabileceğimiz iki kalıp ifade var:

**visibility: hidden;** - Element gizlenir ancak elementin sahip olduğu alan boşluk şeklinde sayfada görünecektir.

- **display: none;** - Element gizlenir ve sahip olduğu alan da sayfada gösterilemez ve yer kaplamaz.

# Block (Kutu) ve Inline (Satırıçi) İfadeler

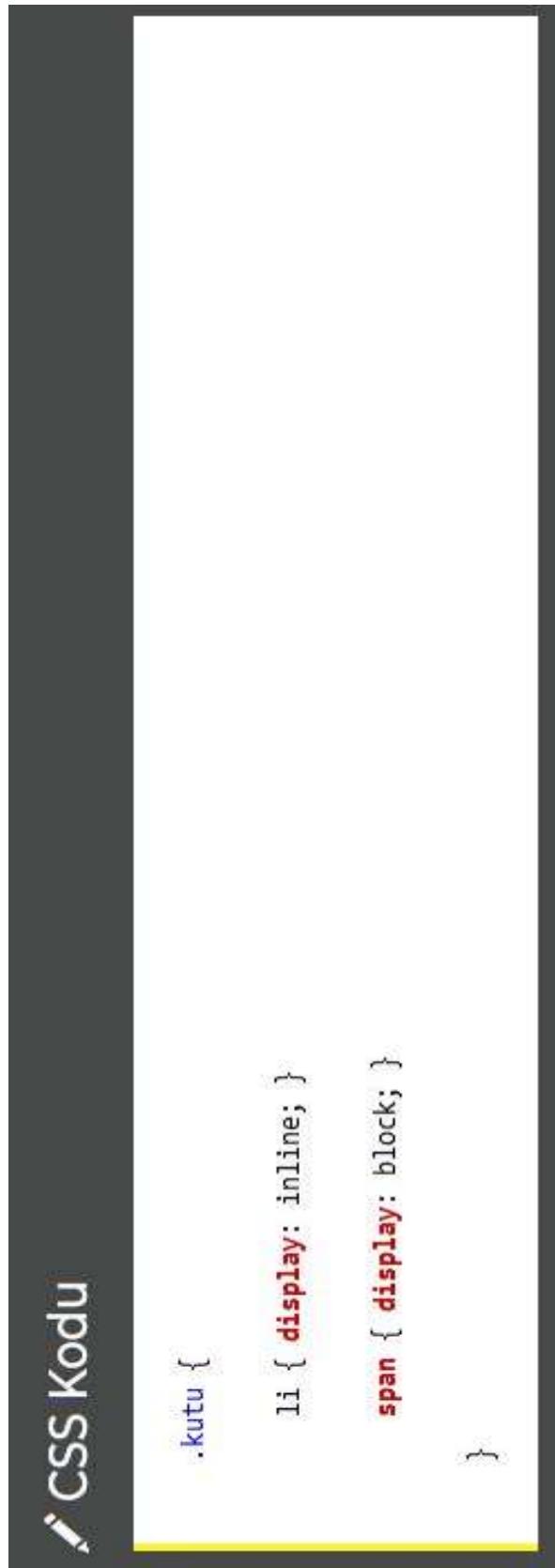
- Bir HTML elementi kutu yapısında olabilir. Bu durumda sayfada bulabildiği tüm genişliği kullanan bir kare alan kaplayacaklardır. DIV, P, H1 elementleri bu tarz elementlerdir.

Bazı elementler ise satır içinde sadece başladığı ve bittiği yere kadar alan kaplar. Bunlarsa satırıçi (inline) elementlerdir.

# Bir Elementin Kutu Ya Da Satır İçi Olmasını Sağlamak

- Element stiline display'ı kullanarak ekleyeceğimiz şu özellikler HTML elementinin kutu mu yoksa satır parçası mı olacağını belirtmemizi ve değiştirmemizi sağlar:
- display: block; - Element kutu Şeklinde alan kaplayacaktır. Kullanıldığı anda yeni bir satırdan başlar.
- display: inline; - Element satır içinde yer alacaktır. Kullanıldığı satırda devam edecektir.

- Element Özelliklerini değiştirmekle ilgili iki örnek verelim. İlk örneğimizde aslında BLOCK (kutu) tarzı olan li elementini satır içi ifadeye dönüştürüyoruz:



The screenshot shows a code editor window with a dark theme. On the left, there is a sidebar with a pencil icon and the text 'CSS Kodu'. The main area contains the following CSS code:

```
.kutu {  
    li { display: inline; }  
  
    span { display: block; }  
}
```

# CSS Pozisyon Belirleme (position)

- CSS aynı zamanda HTML elementlerinin diğer elementlere göre konumunu belirlemenizi sağlar.
- Bir HTML elementini diğerinin altına ya da üstüne alabilir ve bir HTML elementi fazla büyükse ne yapılacağına karar verebilirisiniz.

- Tüm elementler üst (top), alt (bottom), sağ (right) ve sol (left) Özelliklerine sahiptir.
- Ancak bu özellikler CSS'deki position özelliği belirtmemişse işe yaramayacaktır.
- Ayrıca birbirinden farklı dört pozisyon belirleme yöntemi vardır. Bunlar:

- static - Hiçbir özellik belirtilmeye
- fixed - Sabit, tarayıcıya göre hareket etmeyen element
- relative - Normal olması gereken pozisyon'a göre konum belirlemek
- absolute - Koordinat belirleyerek sayfada bir yere yerleştirmek

## static: Olması Gerektiği Gibi Bırakmak

- Eğer **position** CSS kodunu hiç kullanmadıysak HTML nesneleri olduğu gibi sıralanacaktır.
- Bu Şekilde olursa HTML elementinin alt, üst, sağ ve sol özellikleri belirlenemez.

## fixed: Sabitlemek

- Bir HTML elementini tarayıcı ekranına sabitlemek. Kaydırıcı (scroll) kullanısanız bile o element orada yer alacaktır (hareket etmez).

✓ CSS Kodu

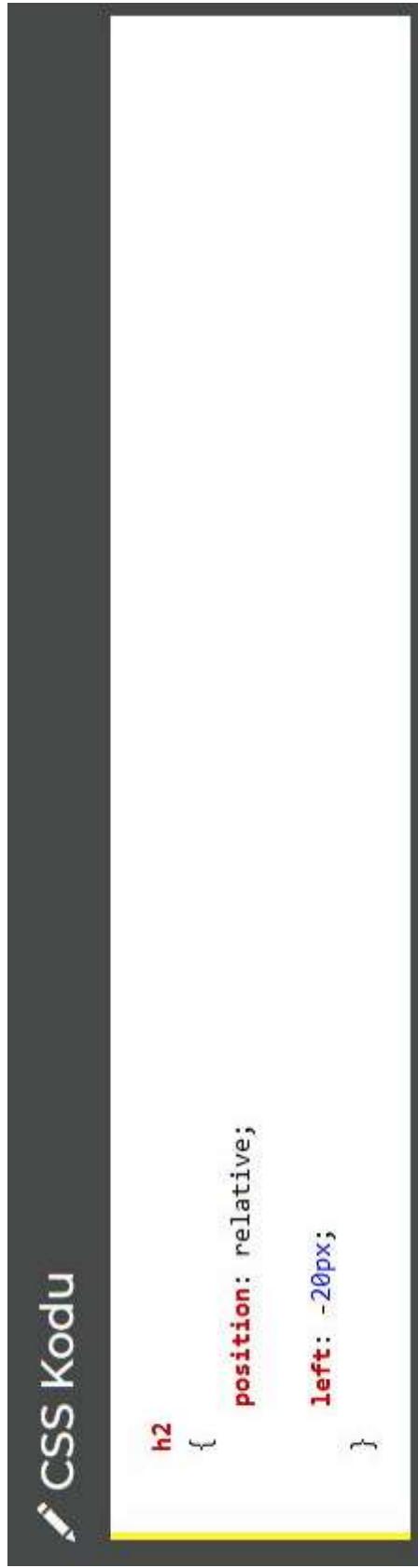
```
p.fixed {  
    position: fixed;  
  
    top: 30px;  
    right: 5px;  
}
```

- Bir önceki sayfadaki kod çalıştırılırsa "fixed" özelliğine sahip P elementinin sağдан 5 piksel üstten 30 piksel noktasında sabit bir Şekilde kaldığını göreceğiz.
- Bu tarz elementler bir diğerinin üzerine geçebilir ve diğerini kapatabilir.

# relative: Normal Pozisyon'a Göre Konumlama

- Bir HTML elementini sahip olduğu pozisyondan yukarı, aşağı, sağa ve sola doğru ayarlamamızı sağlar.
- Elementin kapladığı alan tarayıcı tarafından tutulmuştur, dolayısıyla bunda bir değişiklik olmaz.
- Ancak biz sağ, sol, alt ve üst tarafa doğru hareket ettirebiliriz.

- Aşağıdaki örnekte H2 elementi kullanıldığı yerden 20 piksel sola kayacak ve oradan başlayacaktır. Bu durumda diğer elementlerin alanına girebilecektir.



The screenshot shows a code editor interface with a dark theme. On the left, there is a vertical toolbar with a pencil icon labeled "CSS Kodu". The main area contains the following CSS code:

```
h2
{
  position: relative;
  left: -20px;
}
```

The code is written in a monospaced font, with the selector "h2" in red, the keyword "position" in blue, and the value "relative" in green. The "left" keyword and its value are also in blue. The code is enclosed in curly braces, with a yellow horizontal bar highlighting the opening brace on the right.

# **absolute: Herhangi Bir Konum Belirleme**

- Herhangi bir elemente göre ya da element belirtilmeyse tüm HTML sayfasına göre yer alacağı konum piksel olarak ifade edilir.
- Örneğin H2 elementinin sayfanın başlangıç noktasının (0, 0) 100 piksel sağında ve 150 piksel altında (100, 150) olmasını istiyorsak:

- Bu tarz belirlenen pozisyonlar RELATIVE'den farklı olarak alan kaplamazlar. Yani sayfamızdan soyutlanmışlardır. Bağımsız hareket eden ve sadece tek bir başlangıç noktasını dikkate alan elementler gibi düşünülebilir.

```
h2
{
    position: absolute;
    left: 100px;
    top: 150px;
}
```

# **z-index: Elementin Öncelliğini Belirlemek**

- Nasıl Photoshop tarzı programlarda katmanlarla çalışılıyor ve katmanların sırası belirlenebiliyorsa, CSS'de de z-index özelliği ile bunu belirleyip bir HTML elementinin diğerinin üzerinde ya da altında görünmesini sağlayabiliyoruz.

```
img  
{  
    position: absolute;  
  
    left: 0px;  
    top: 0px;  
  
    z-index: -1;  
}
```

- Yukarıda -1 değeri ile belirttiğimiz resim diğer tüm elementlerin altında görünmeyecektir. Çünkü diğer elementlerde bu değer belirtilmemiye 0 (sıfır) olarak algılanacaktır. Buna karşın bu element -1 olduğu için alt katman kabul edilecektir.

## Ek not!

- Negatif değerler en alt katmanı pozitif değerler üst katmanları ifade eder. Örneğin biri -2 biri -5 olan iki z-index Özelliğine sahip elementten daha büyük olan -2 Özelliğine sahip element daha üstte görünecektir.

# CSS Kaydırma (float)

- Katmanlarla çalışmak ve resim galerileri oluşturmak için HTML elementlerini sağa ve sola doğru kaydırımız gerekebilir. İşte css kodu float bunu yapmamız sağlıyor.

# Elementler Nasıl Kaydırılır?

- Elementler yatay yönde sağa ve sola doğru kaydırılabilir. Yani sağa doğru kaydırma söz konusu değil.

Kaydırılan element, sağ ya da soldaki elementin sağından veya solundan başlar. Yeni kaydırılan elementler, ilk önce kaydırılan elementten sonra gelir. Dolayısıyla önceden kaydırılmış elementlerde bir değişiklik olmaz.

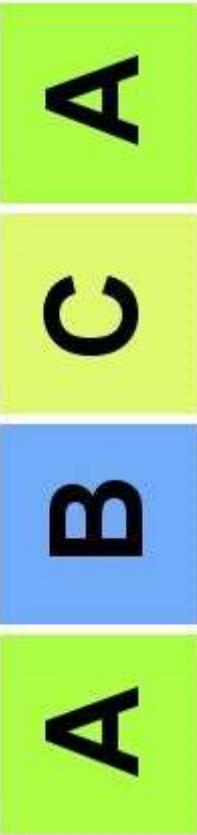
Örneğin bir resim için float: right; özelliği belirttiysek yazılar sağ tarafta yer alacak bu resmin solunda yer alacaktır.

# Elementleri Yan Yana Sıralamak

- Resim galerilerini görmüşsünüzdür. Resim sayfalarına giden küçük görüntüler sayfada yan yana sıralanırlar. Bunlar birbirine eşit olmalı ki kaydırıldığında birbiriyle uyumlu olsunlar. Örneğin küçük görüntümüz.

css Kodu

```
.thumb {  
    float: left;  
  
    width: 110px;  
    height: 110px;  
  
    margin: 5px;  
}
```

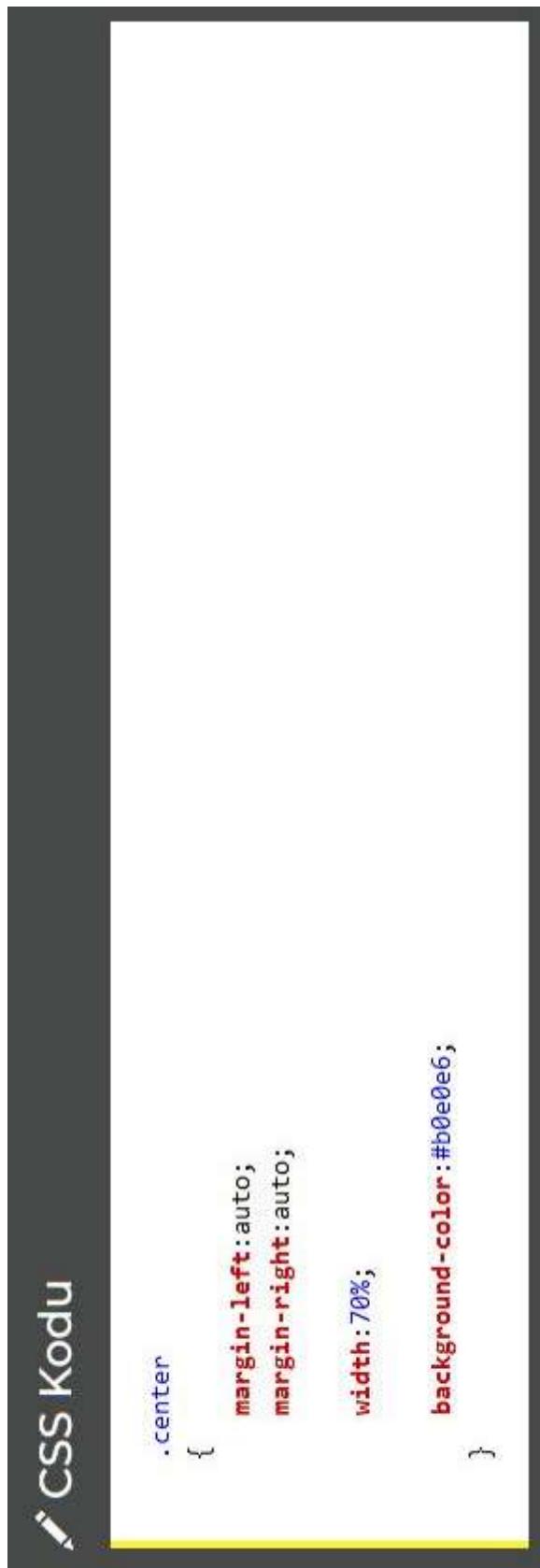


- Yukarıdaki örnek kodda yer alan Özelliğe sahip resimleri sıraladık. Görüldüğü gibi her biri diğerinin sağına kaydırıldı ve aralara  $5 \times 2$  (sağ, sol) piksellik boşluk (margin) verildi. Sığmayan nesneler ise bir aşağı satırı kaydırıldı.

# CSS Hızalama (align)

- Kutu Özelliği Elementlerin Hızalanması
- Kutu Özelliği nesnelerde hizayı ayarlamak için text-align kullanılır.
- Margin Kullanarak Ortalamak
- Kutu Özelliği elementler sağ ve sol dış boşluk (margin) değerleri auto yapılarak ortalanabilir. Tarayıcı bu durumda sağ ve sol boşlukları kendi ayarlayacağı için yazı ortalanacaktır.

- Ancak bu durumda genişlik yani **width** hiçbir zaman 100% olmamalıdır. Aksi takdirde kenarlık tarayıcı tarafından ayarlanamaz.  
Örneğe balkalım.



The screenshot shows a dark-themed code editor window titled 'css Kodu'. Inside the editor, there is a single line of CSS code:

```
.center {  
    margin-left: auto;  
    margin-right: auto;  
    width: 70%;  
    background-color: #b0e0e6;  
}
```

The code uses standard CSS properties like margin, width, and background-color. The background color is set to a light blue (#b0e0e6). The code is enclosed in a pair of curly braces {}, indicating a CSS selector block.

- Bir önceki sayfadaki örnek örnekteki tarayıcı otomatik olarak 15% sola 15% sağa boşluk verir ve genişliğe göre bunu oranlar.
- **Position Kullananak Sağ'a ve Sola Yaslamak**
- Normal akıştan farklı olacak bir position: absolute; kodu kullanarak belirttiğimiz genişlikte bir kutunun sağa ya da sola yaslanmasını sağlayabiliriz. Örneğin 500 piksellik bir kutuyu sağa yaslamak isteseydik:

Sağdan 0 piksel boşluk bırakılacağı  
right: 0px; ile belirtmiş olduk.

/ CSS Kodu

```
.center {  
    position: absolute;  
    right: 0px;  
    width: 500px;  
    background-color: #b0e0e6;  
}
```

# Float Kullanarak Sağ'a ve Sola Yaslamak

- Float HTML elementlerini sağa veya sola kaydırarak listelememizi sağlar. Kaydırma özelliğini hizalamak için kullanabiliriz.

 CSS Kodu

```
.right
{
    float: right;
    width: 300px;
    background-color:#b0e0e6;
}
```

# Element Özelliklerine Göre Stilleme

- HTML dilinde elementler belli özellikler alabilmektedir. Örneğin HREF bir A elementi özelliğidir. İşte buradaki element özelliklerine göre stillemeye yapmamız mümkün. Sadece iki özelliğe göre bunu yapamayız. Bunlar id ve class. Şimdi örneğe bakalım. Aşağıdaki kod title özelliğine sahip tüm elementlerin kırmızı renkli olacağını bildirir.

css Kodu

```
[title]
{
    color: red;
}
```

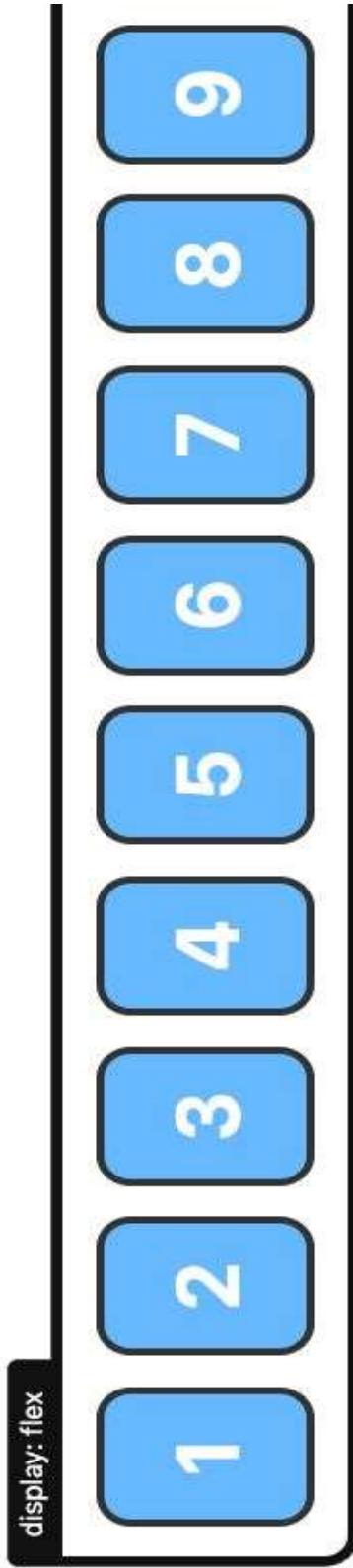
# CSS FLEX YAPISI

- flex yapısı ile, artık kapsayıcıya ve içindeki öğelerine esneklik getirebiliyoruz.
- Flex'i kullanmadan en büyük sebebi, esnek yapıları kolayca yönetmek için. Çünkü yapıya esneklik verdiği için, yatay ve dikey hizalarda nasıl görüneceğini, öğelerin kendi içinde hizalanmalarını ve sırasını belirlemek gibi güzel özellikleri bulunuyor.
- Ayrıca tüm çözümlüklerde ve cihazlarda daha hızlı ve esnek bir yapı kullanmak için, flex bizim tama da ihtiyacımız olan şey.

- Flex genel anlamıyla bir CSS Özelliğinin adı değil, yapının adıdır.
- Dolayısı ile flex yapısı altında hem kapsayıcı (container) için hemde içindeki öğeler (items) için birden fazla flex Özelliği bulunmaktadır.
- Bu çalışmada hepsini incelemeye ve anlamaya çalışacağız.

# Kapsayıçı (Container) için Flex Özellikleri

- Flex modülü hem kapsayıçı hem de öğeler için özellikler sunuyor. İlk olarak kapsayıçı için hangi özellikler var, nasıl kullanılıyor bunlara bir gözatalım.



```
.container {  
    display: flex; /* ya da inline-flex */  
}
```

- **display: flex;** ya da **display: inline-flex;** kullanılarak kapsayıcı içerisindeki öğelere artık esneklik vermeye başlayabiliriz
- Bu tanımla artık bu kapsayıcının esnek bir görünümde olduğunu belirtiyoruz.
- **flex** ile **inline-flex** farkı, **block** ile **inline** tanımları ile aynıdır.

# Flex Yönleri

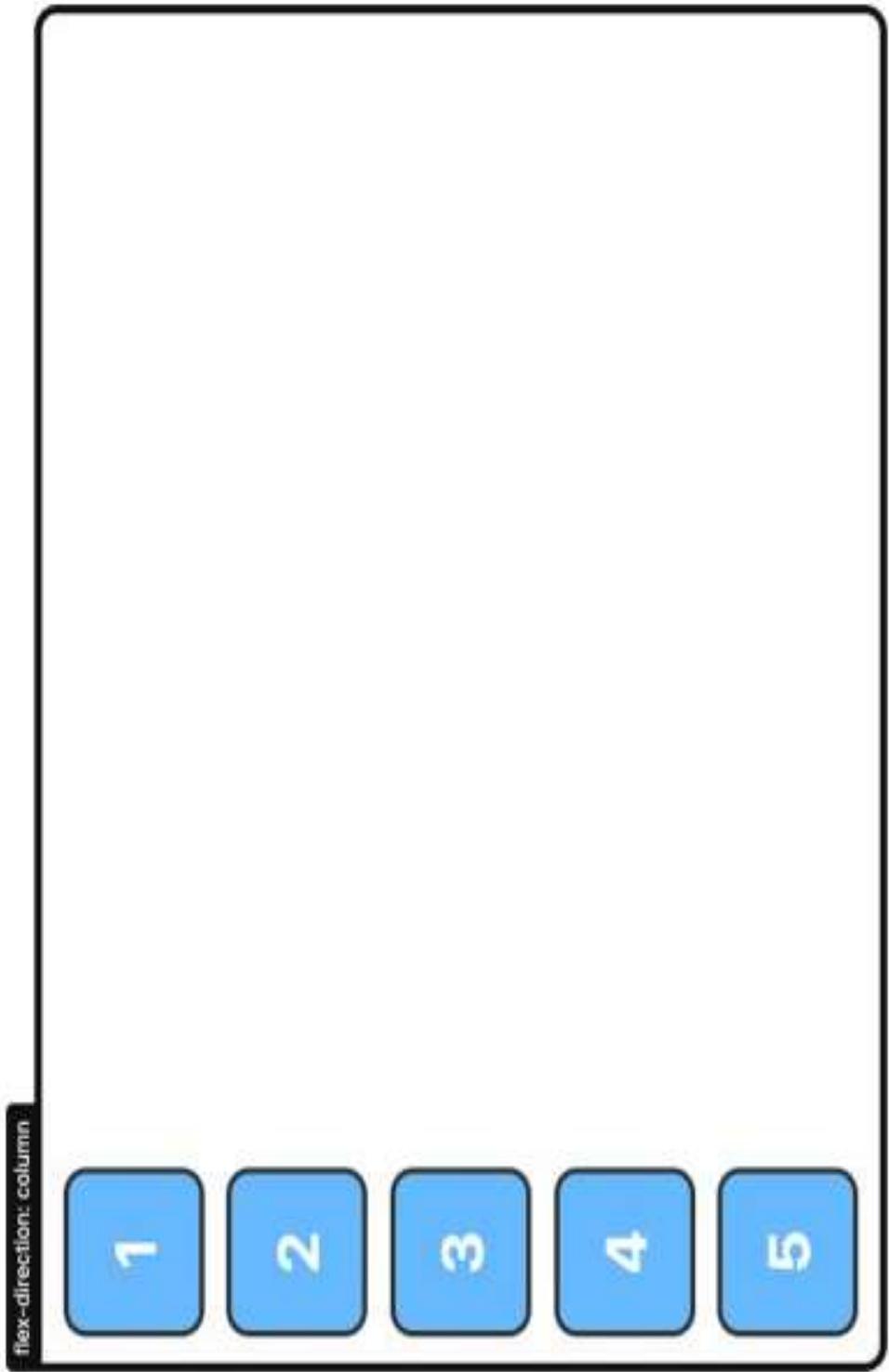
flex-direction: row;

- 1
- 2
- 3
- 4
- 5

flex-direction: row-reverse;

- 1
- 2
- 3
- 4
- 5

# Flex Yönleri 2

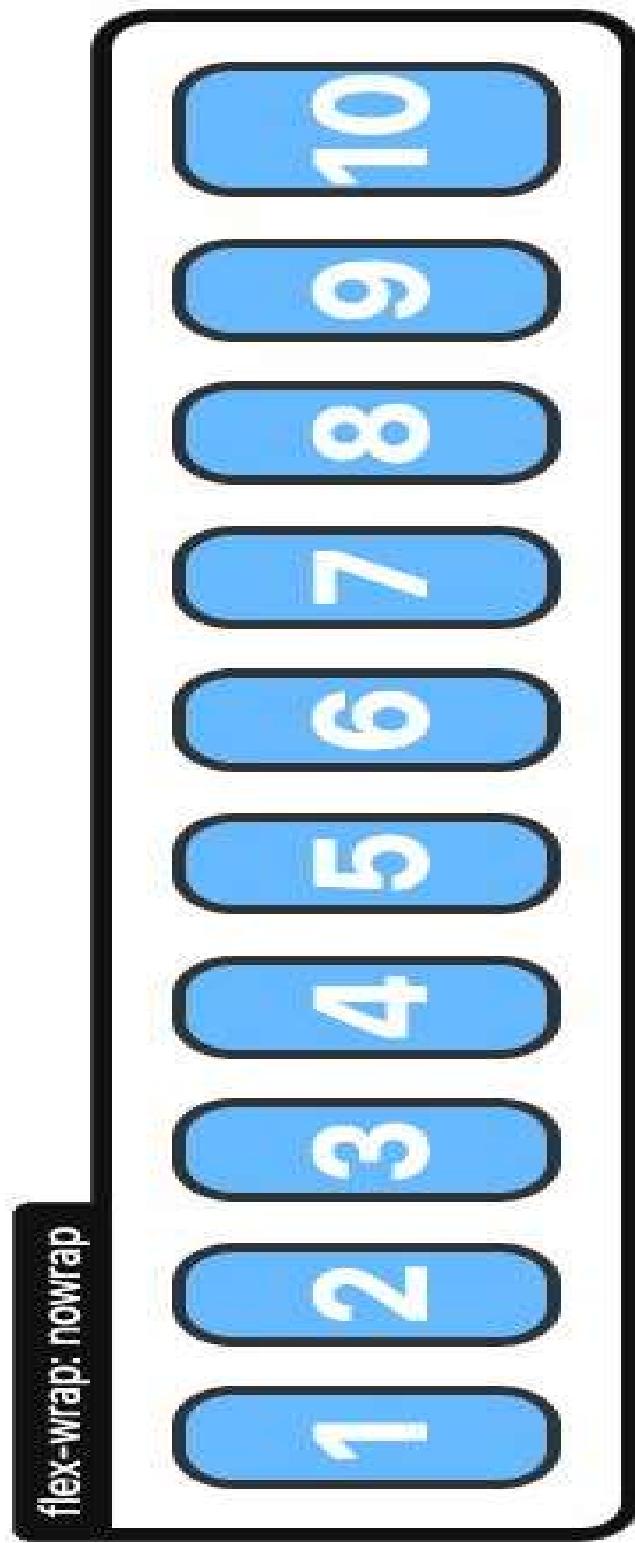


- Flex Direction öğelerin yönünü belirlemek için kullanılır.
- Varsayılan olarak flex container yataya doğru uzar.
- Dolayısı ile bu kapsayıcı içindeki tüm öğeler yan yana listelenir.
- Bu Özelliğin aldığı birkaç değeri inceleyip ne işe yaradıklarını anlayalım;

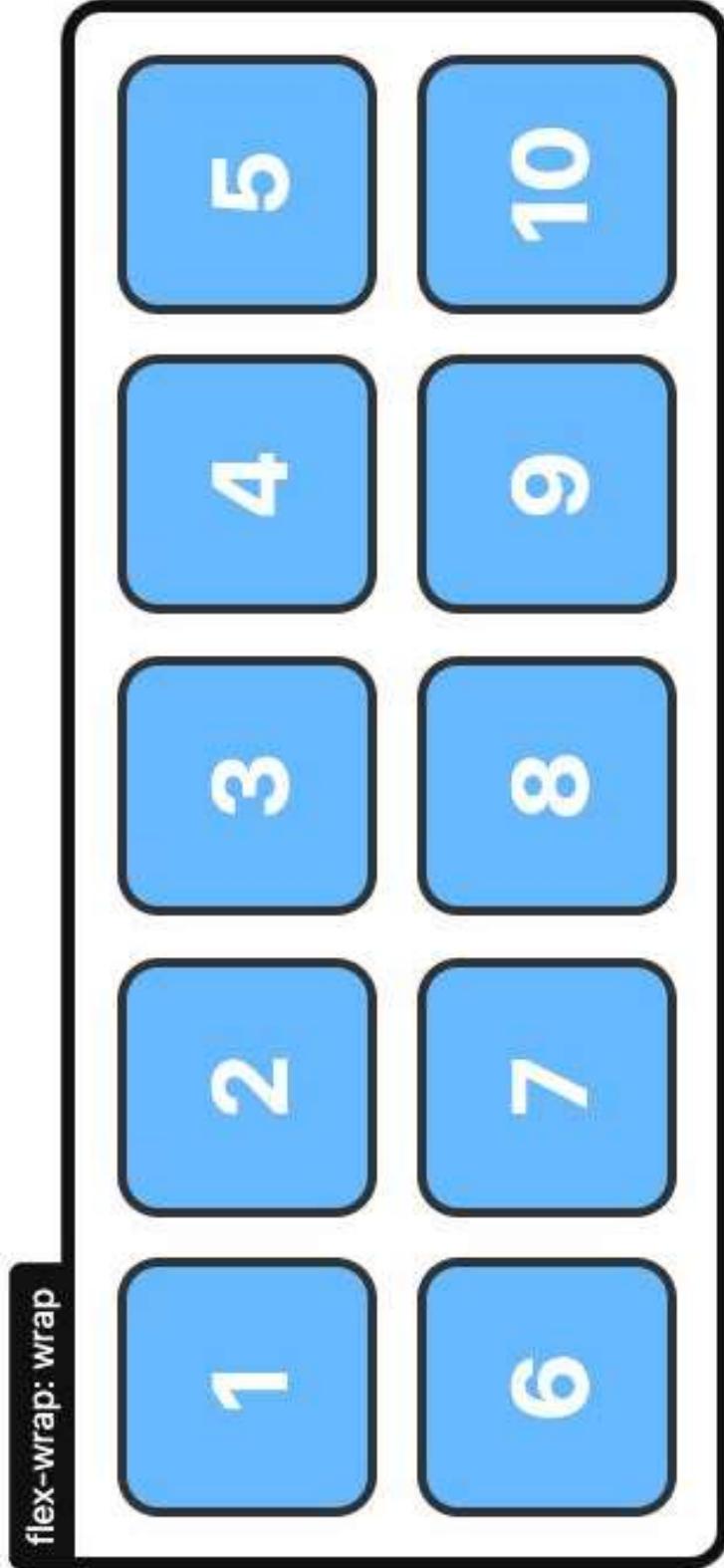
```
.container {  
    flex-direction: row | row-reverse | column | column-reverse;  
}
```

- `row` (`varsayılan`) = soldan sağa doğru sıralar.
- `row-reverse` = sağdan sola doğru sıralar.
- `column` = yukarıdan aşağı doğru sıralar.
- `column-reverse` = aşağıdan yukarı doğru sıralar.

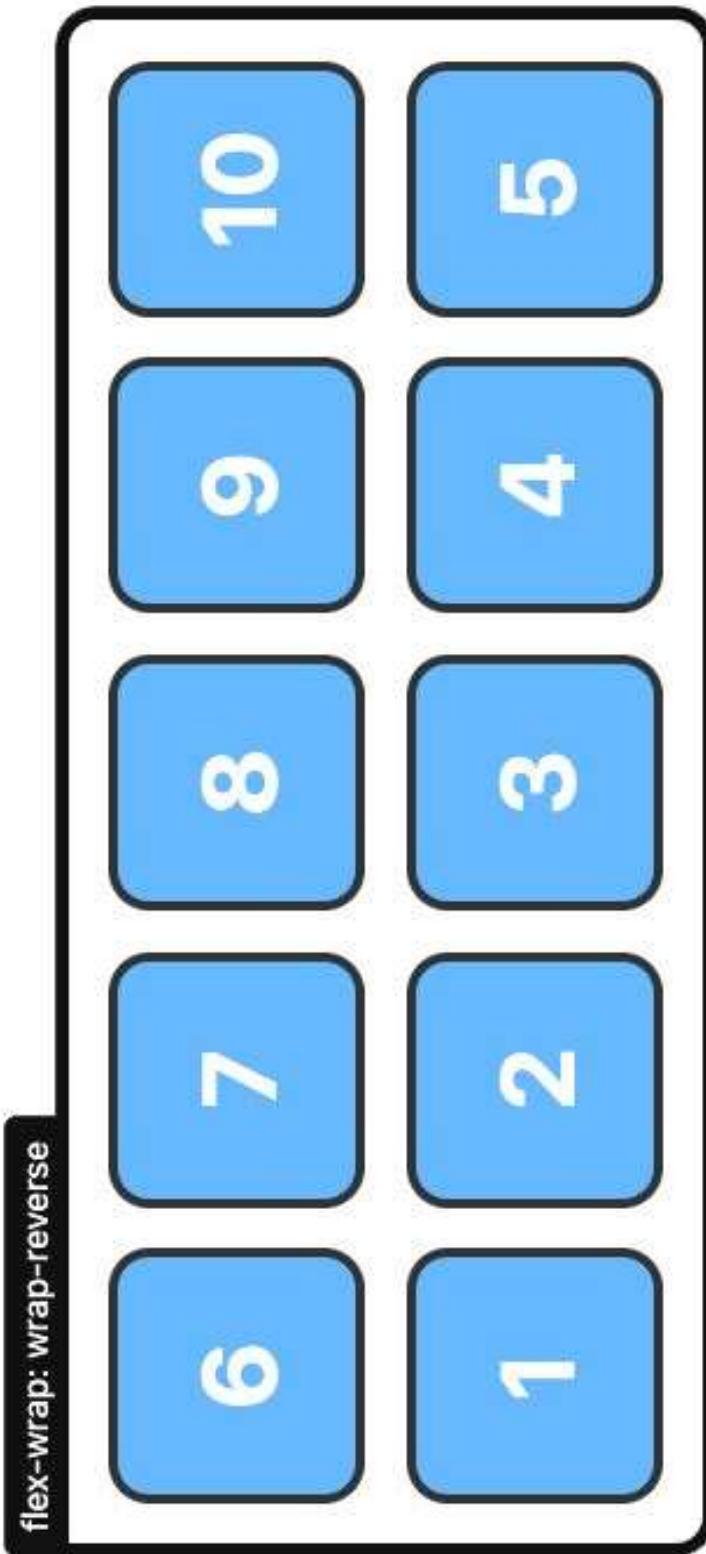
# Flex Wrap Özellikleri



# Flex Wrap Özellikleri 2



# Flex Wrap Özellikleri 3

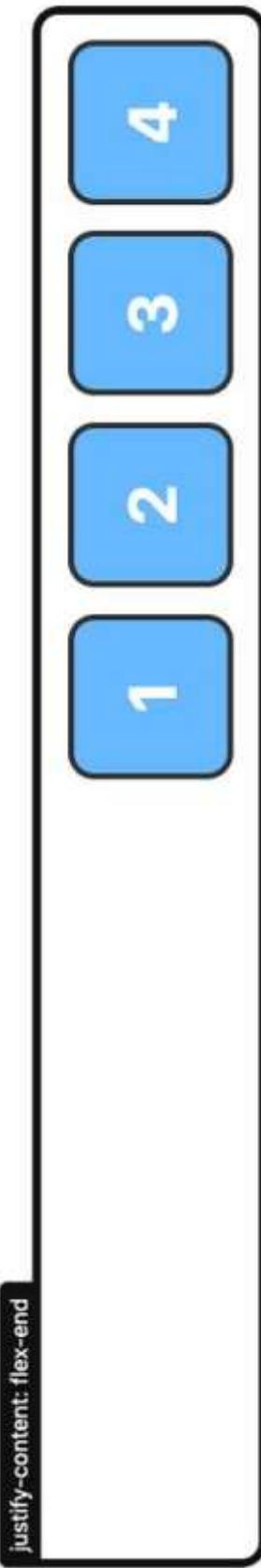
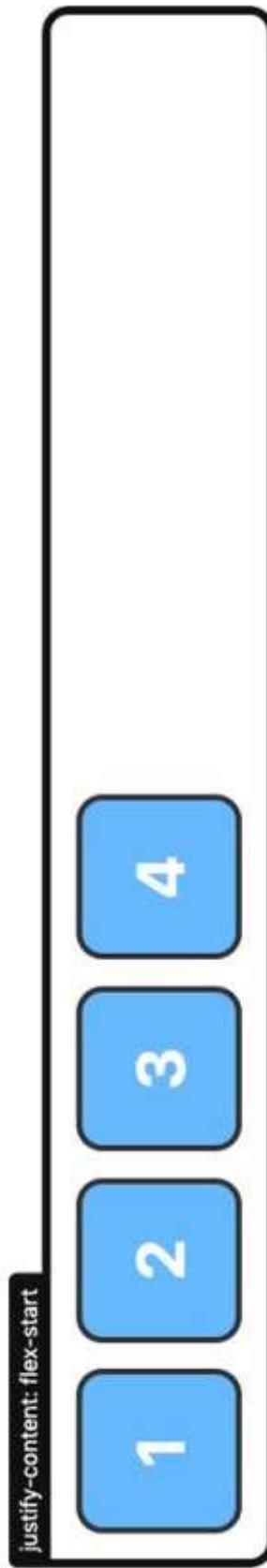


- Varsayılan olarak esnek öğeler tek bir satıra sızmaya çalışırlar. Flex Wrap gerektiğiinde birden fazla satıra yaymak için bu özellik kullanılır.

```
.container {  
    flex-wrap: nowrap | wrap | wrap-reverse;  
}
```

- nowrap (varsayılan) = Tüm öğeleri tek bir satıda yan yana listeler.
- wrap = Öğeler gerektiğiinde birden fazla satırda yukarıdan aşağı doğru listelenir.
- wrap-reverse = Öğeler gerektiğiinde birden fazla satırda aşağıdan yukarı doğru listelenir.

# Justify Content Özellikleri



# Justify Content Özellikleri 2

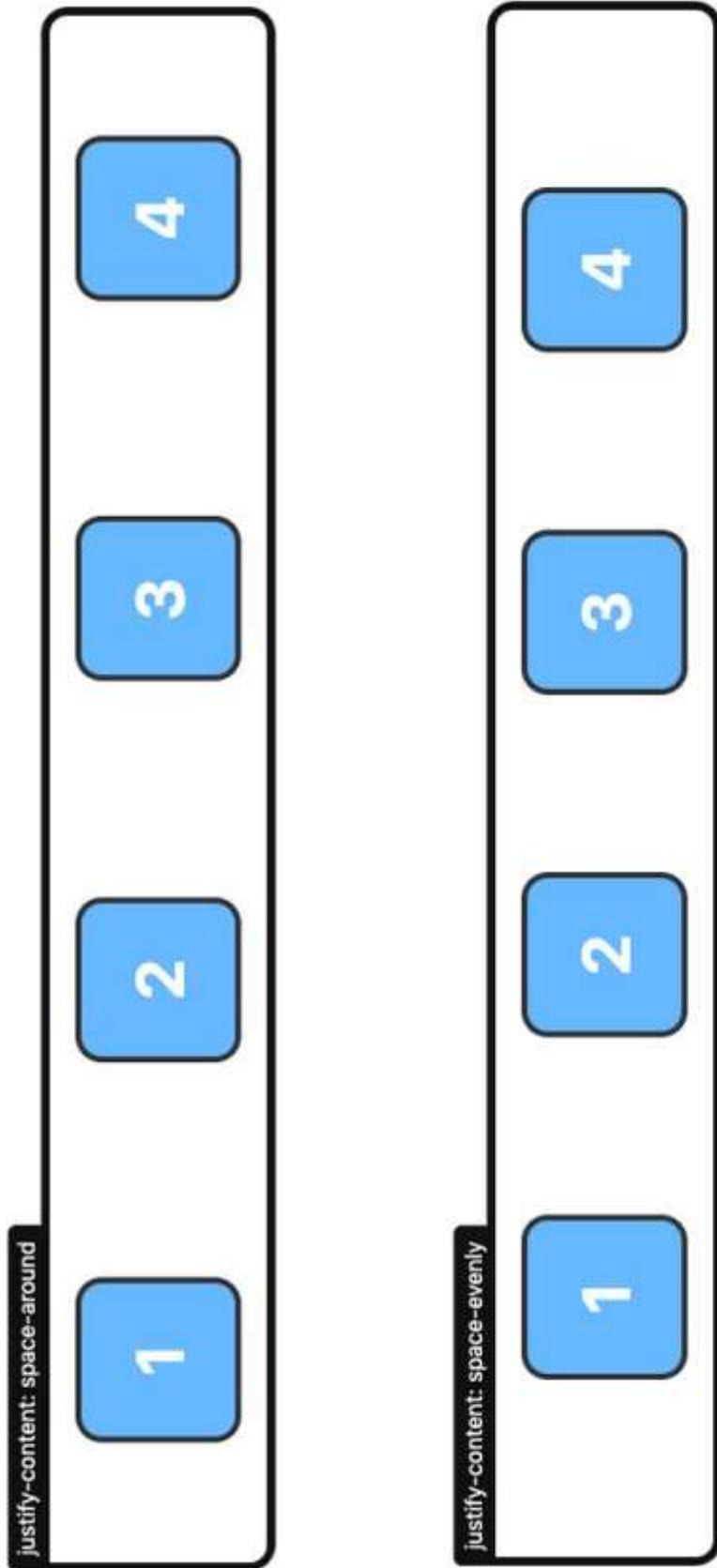
justify-content: center

- 1
- 2
- 3
- 4

justify-content: space-between

- 1
- 2
- 3
- 4

# Justify Content Özellikleri 3



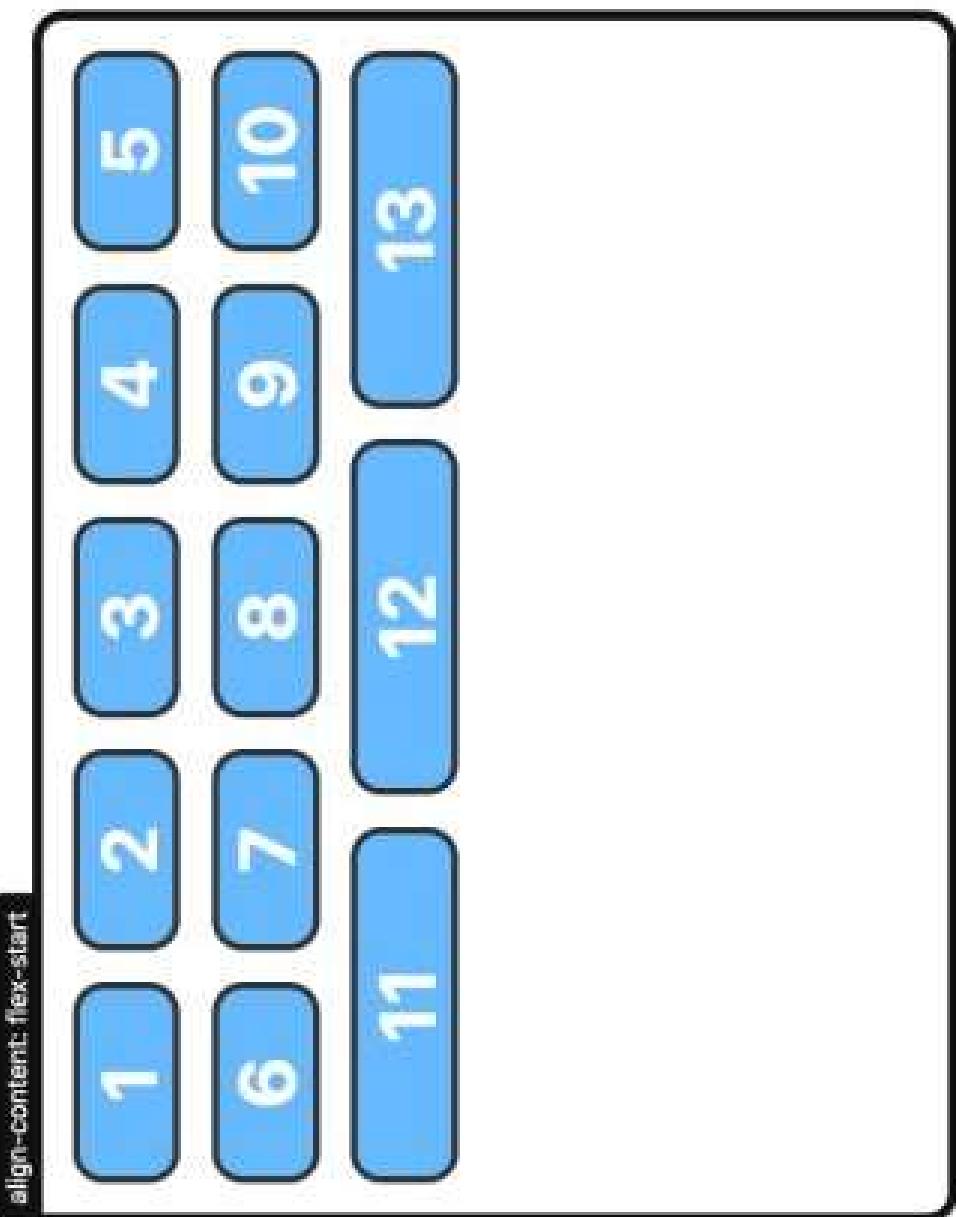
```
.container {  
    justify-content: flex-start | flex-end | center | space-between | space-around | space-even  
}
```

- Öğelerin yatay eksende hizasını ayarlamak için kullanılır. Yukarıdaki görselde kullanımları ve sonuçları gösterilmiştir. Birde değerlererin ne işe yaradıklarına bakalım.
- flex-start (varsayılan) = Öğeleri başlangıç noktasında (sola dayalı) hizalar.
- flex-end = Öğeleri bitiş noktasında (sağa dayalı) hizalar.

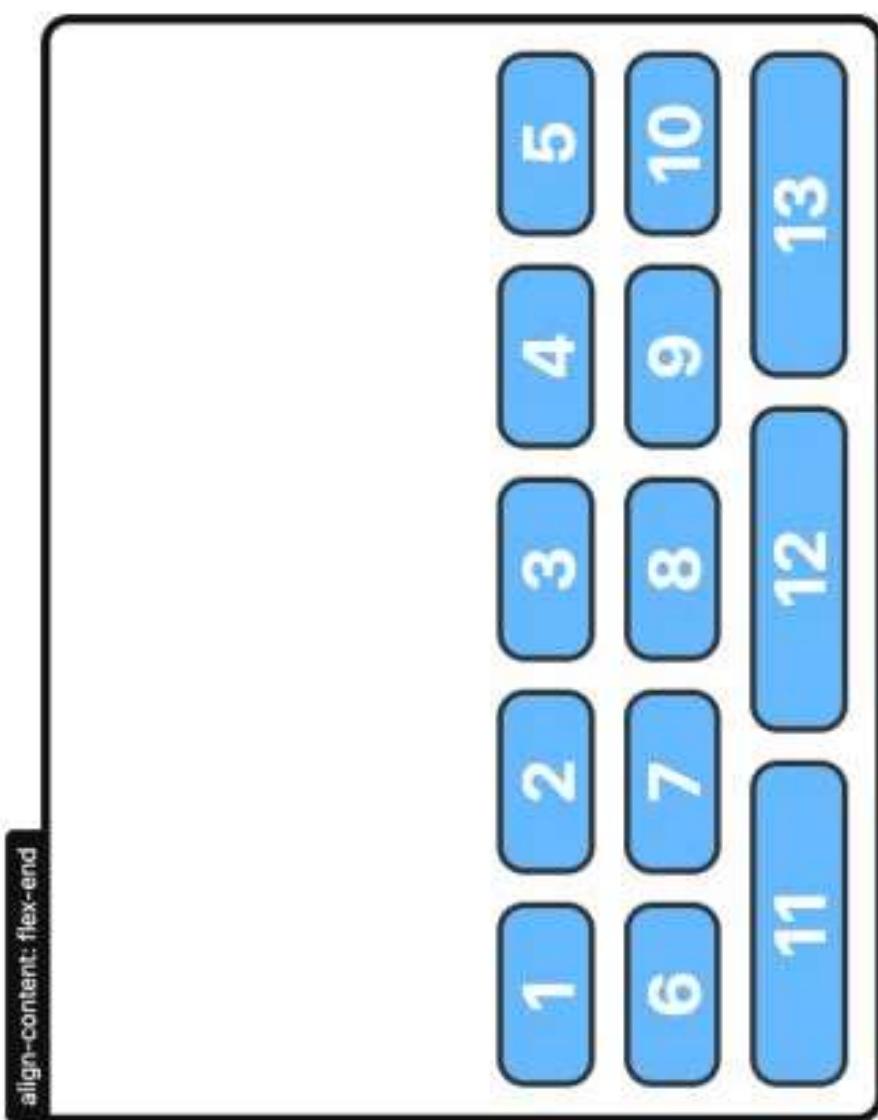
- center = Öğeleri ortada hizalar.
- space-between = İlk öğe sola dayalı, son öğe sağa dayalı ve diğerleri orta olacak şekilde hizalar.
- space-around = İlk ve son öğe hariç diğerlerinin arası eşit olacak şekilde hizalar.
- space-evenly = Tüm öğelerin arası aynı eşitlikte olacak şekilde hizalar.

# Align Items Özellikleri

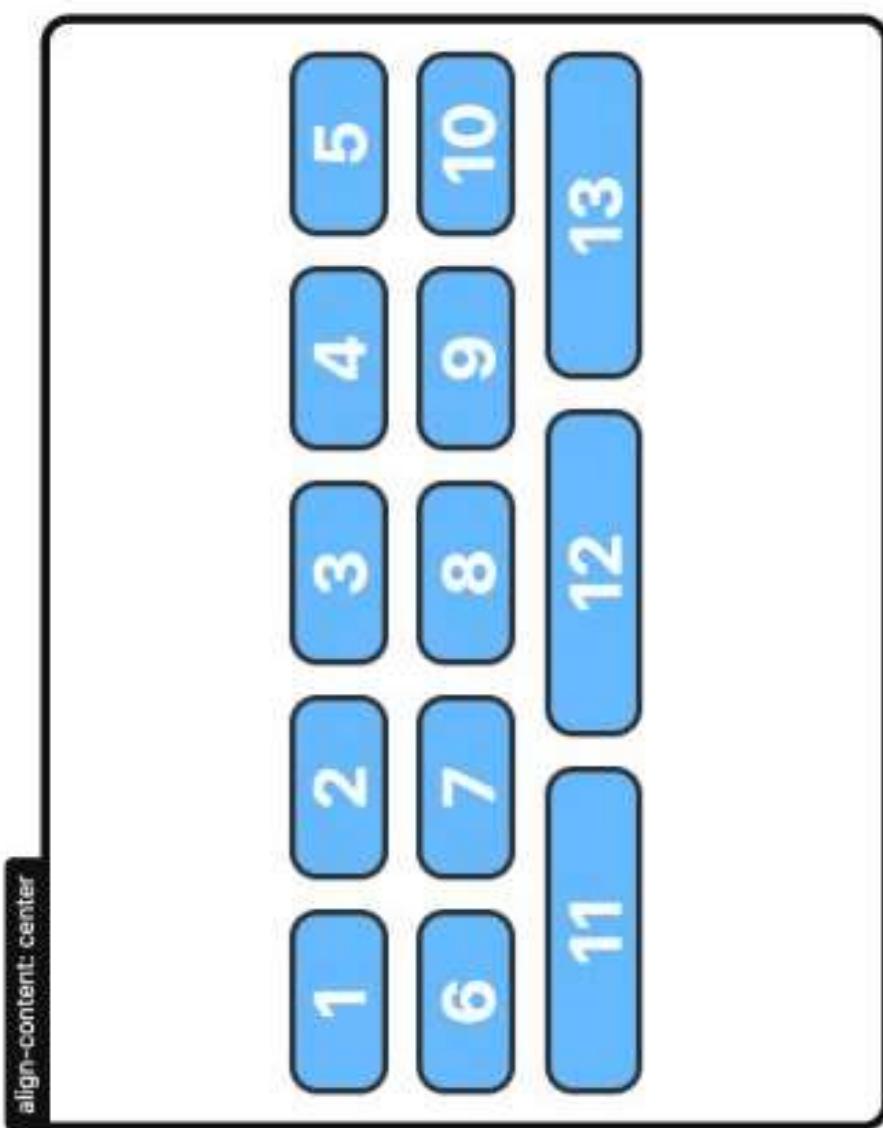
align-content: flex-start



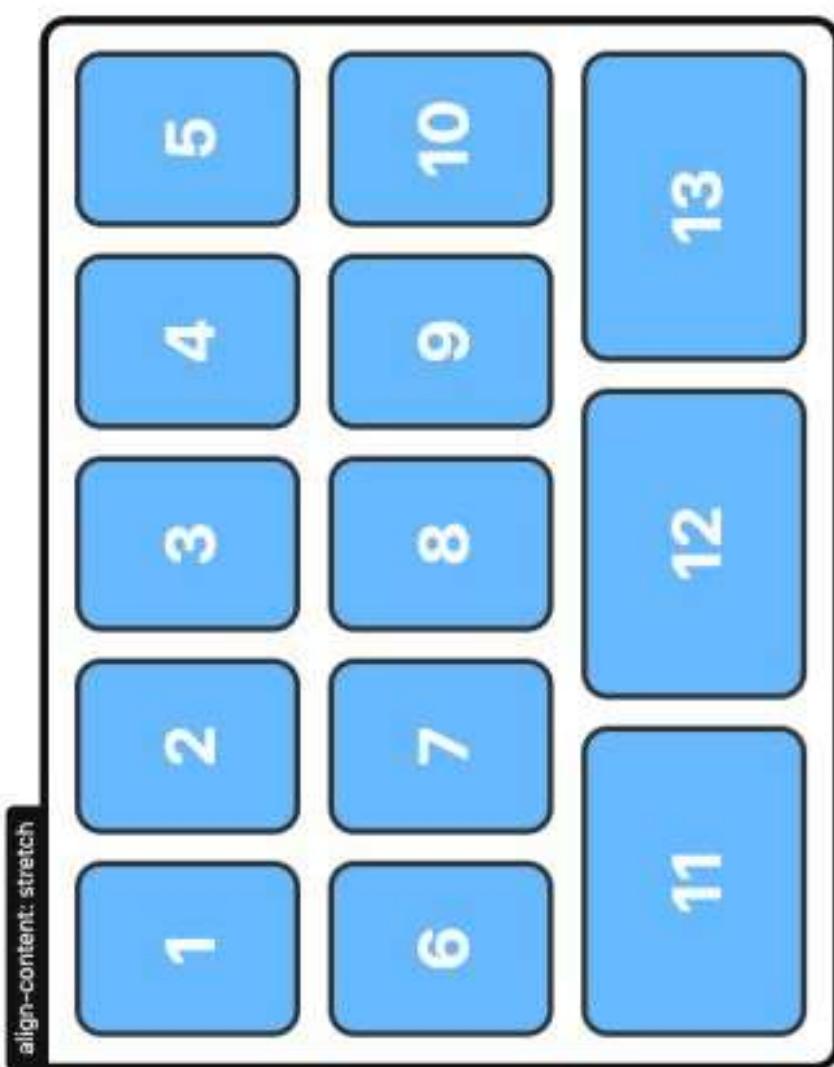
# Align Items Özellikleri 2



# Align Items Özellikleri 3

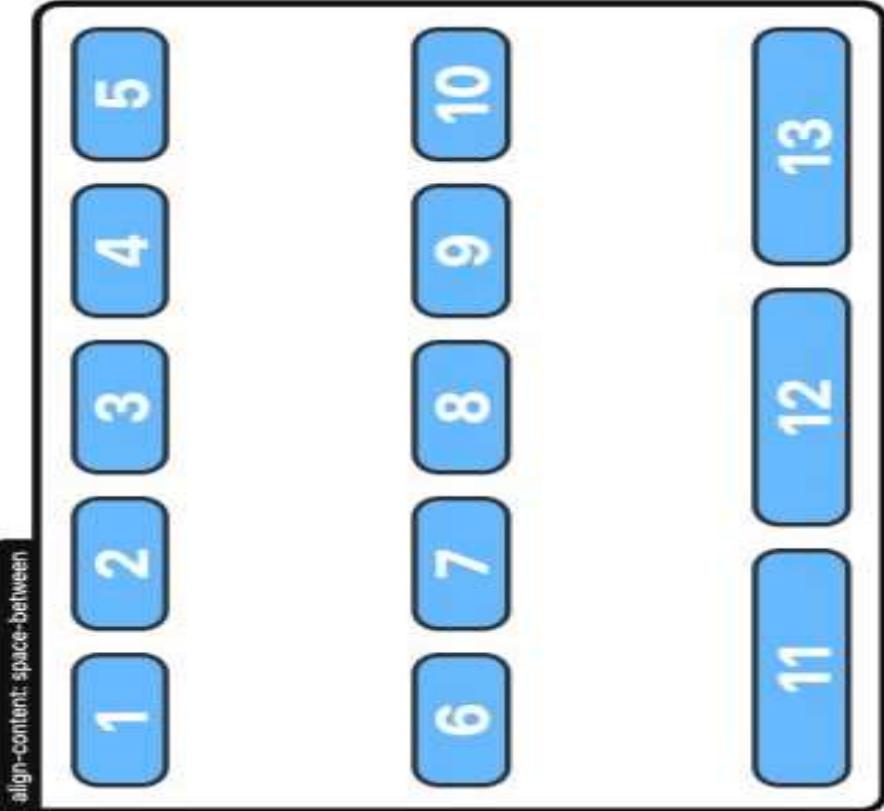


# Align Items Özelliği 4

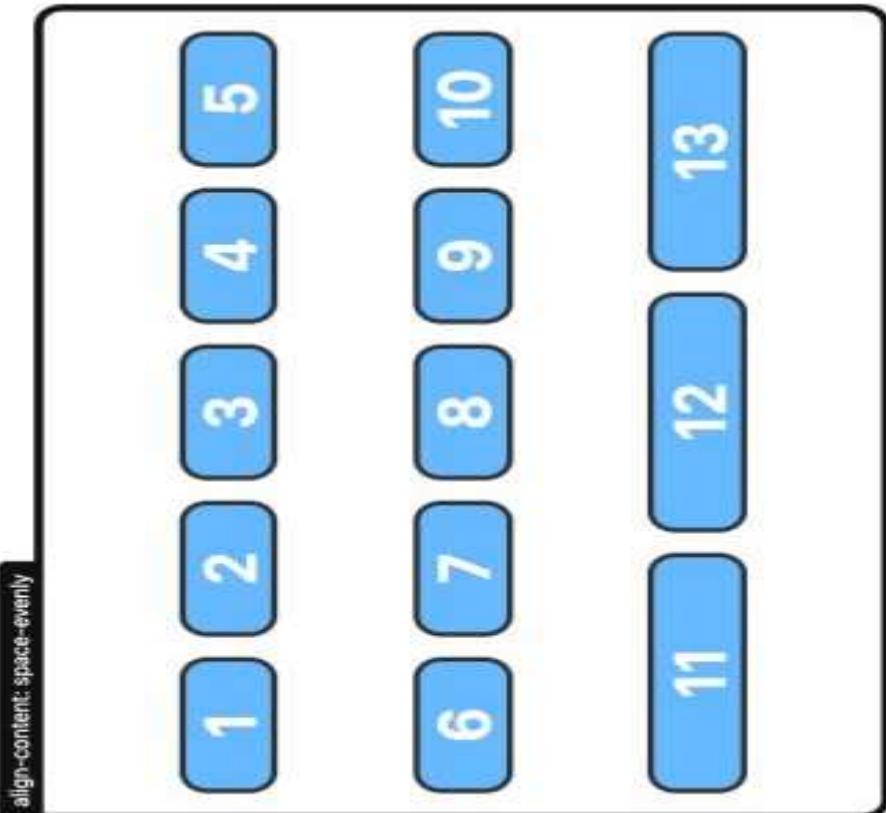


# Align Items Özellikleri 5

align-content: space-between



align-content: space-evenly



```
.container {  
    align-content: stretch | flex-start | flex-end | center | space-between | space-around;  
}
```

- Çok satırlı yapılarda öğelerin dikey eksende hızmasını ayarlamak için kullanılır.
- Yukarıdaki görselde kullanımları ve sonuçları gösterilmiştir.
- Birde değerlerin ne işe yaradıklarına bakanız.

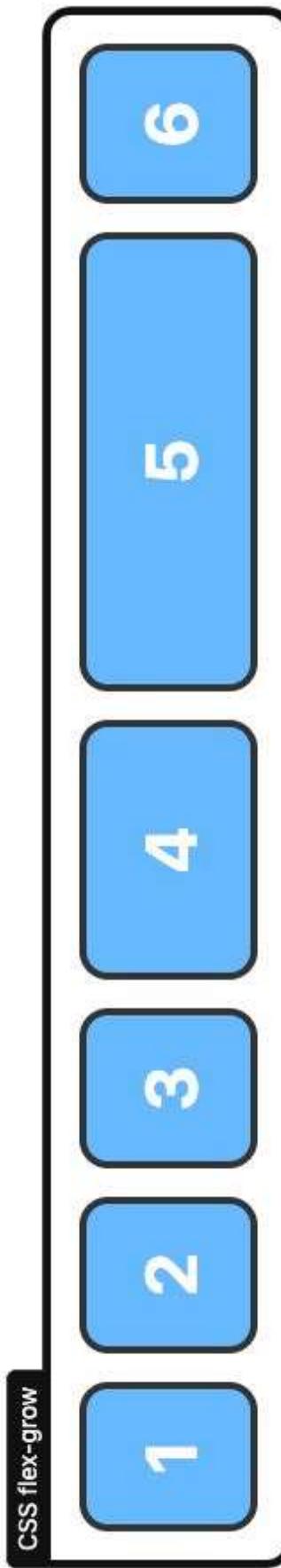
- flex-start = Öğeleri dikey eksende başta hizalar.
- flex-end = Öğeleri dikey eksende sonra hizalar.
- center = Öğeleri dikey eksende ortalı hizalar.
- stretch (varsayılan) = Öğeleri hizalamaz, yüksekliklerini uzatır.

- space-between = İlk satırdaki öğeler başta, son satırdaki öğeler sonra olmak üzere diğer satırdaki öğeleri dikey eksende ortalar.
- space-evenly = Öğeleri satırlar arası eşit oranda olacak şekilde dikey eksende ortalar.

# Flex Grow

- Bir öğeyi diğerlerine göre daha büyük göstermek için bu özelliği kullanıyoruz.
- Varsayılan değeri 0'dır.
- Eğer öğelerden birisinin değerini 2 yaparsanız diğerlerine göre daha büyük olacaktır ancak kalanı daralacaktır.

- İlk olarak tüm öğelerin değerini 1 yapıp, daha sonra bir öğenin değerini 2 yaparsanız, sonuç olarak diğer öğelere göre 2 kat daha büyük bir öğeye sahip olacaksınız.



```
.item {  
  flex-grow: 1;  
}  
.item4 {  
  flex-grow: 2;  
}  
.item5 {  
  flex-grow: 4;  
}
```

# CSS DERS VIDEOLARI

- Türkçe Kaynak:  
<https://www.youtube.com/watch?v=sUWEuDSD7bE&list=PLXuv2PShkuHx5O13uWEJ7iwTG2YTuz7u>
- İngilizce Kaynak:
  - <https://www.youtube.com/watch?v=0W6qz0-aDaM&list=PL0Zuz27SZ-6Mx9fd9elt80G1bPcySmWit>