

# HTML Syntax'ı

- **<html>** tagleri arasında yazılır.
- 2 bölümden oluşur: **head ve body**
- **<head>** tag'i içine yazılan şeyler, html dosyası hakkında browsera bilgi verir.

Bu tag içine yazılanlar browserda görünmez.

- **<body>** tag'i içine yazılanlar browser'da görüntülenir.

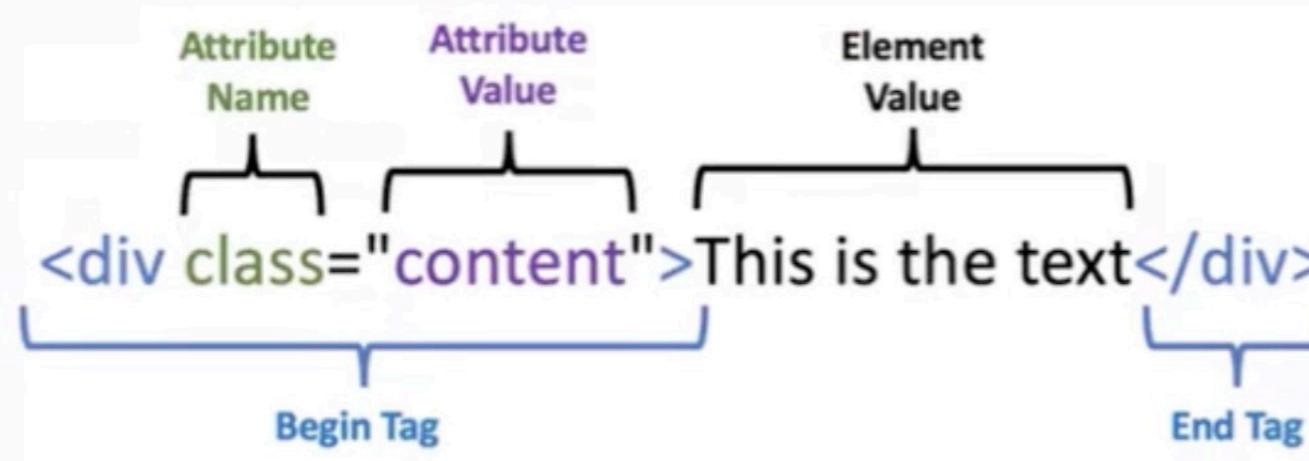
```
<html>
  <head></head>
  <body></body>
</html>
```

```
<html>
  <head></head>
  <body>
    <h1>İstanbul</h1>
    <p>İstanbul Türkiye'de Marmara Bölgesi'nde yer alan şehir ve Türkiye'nin 81 ilinden biridir. Ülkenin nüfus bakımından en çok göç alan ve en kalabalık ilidir. Ekonomik, tarihi ve sosyo-kültürel açıdan önde gelen şehirlerden biridir. Şehir, iktisadi büyülük açısından dünyada 34. sırada yer almaktadır.</p>
    <h2>Tarihi yerler</h2>
  </body>
</html>
```

# HTML Tag'leri

- < ... > içine yazılır. </ ... > şeklinde kapatılır. Örn: <h1>İstanbul</h1>
- Bazı tag'ler **self close**'dur. Ayrıca kapanış tag'i yoktur. sonuna / eklenir. Örn: 
- Temel HTML tag'leri: **h1, h2, h3 ... h6, p, ul, ol, li, a, img, div, span**
  - **a:** anchor tag, sayfalara link vermek için,
  - **img:** sayfadaki resim göstermek için,
  - **ul ve ol:** sayfada liste halinde metin göstermek için,
  - **h1 ... h6:** sayfadaki başlıklar için,
  - **p:** paragraf: sayfadaki metinler için,
  - **div ve span:** sayfadaki elementleri gruplamak için kullanılır

# Tag Attribute'ları



- Attributelerin çoğu tag'e özgürdür.
  - a için href `<a href="https://tr.wikipedia.org">Wikipedia</a>`
  - img için src ``
- Her tag ile birlikte kullanılabilen attribute'lara **Global attribute**'lar denir. Bunlardan en önemlileri id, style ve class 'tir.
- Tag'lere birden fazla attribute eklenebilir.

```
<a target="_blank" href="https://tr.wikipedia.org/">Wikipedia</a>
```

# CSS: Cascading Style Sheets Syntax

```
Selector  
p {  
    color: red;  
}  
      _____  
     |       |  
     | Property   Property value  
     |  
     |_____ Declaration
```

3 farklı yerde yazılabılır:

- inline-style olarak **style** attribute'u ile tag'e
- head bölümünde **<style>...</style>** tag'leri arasında
- farklı bir dosyada yazıp,

head bölümünde html sayfaya **link**lenerek

```
<link rel="stylesheet" href="style.css" />
```

```
<h1 style="color: brown; font-size: 36px;">İstanbul</h1>
```

```
<head>  
    <style>  
        h1 {  
            color: blue;  
            font-size: 36px;  
        }  
    </style>  
</head>
```

# CSS Seçicileri

Tag adları ile seçmek:

```
<h1>İstanbul</h1>
```

```
h1 {  
    color: blue;  
    font-size: 36px;  
}
```

id attribute'u ile seçmek

```
<a id="marmara-link" href="https://tr.wikipedia.org/wiki/Marmara_B%C3%B6lgesi">Marmara Bölgesi</a>
```

```
#marmara-link {  
    text-decoration: none;  
}
```

class attribute'u ile seçmek

```
<h1 class="title">İstanbul</h1>  
<h2 class="title">Tarihi yerler</h2>
```

```
.title {  
    color: purple;  
}
```

# Extra

1. HTML tag'leri: <https://www.w3schools.com/tags/>
2. CSS selector'lari: [https://www.w3schools.com/cssref/css\\_selectors.php](https://www.w3schools.com/cssref/css_selectors.php)
3. CSS propertyleri: <https://www.w3schools.com/cssref/index.php>

# CSS Çalışma Prensibi

## Inheritance: Kalıtım

- HTML elemanları, kendilerini kapsayan elemanların yazı rengi, boyutu, satır yüksekliği, karakter kalınlığı gibi yazı ile ilgili stillerini miras alır.

## Cascading: Katman

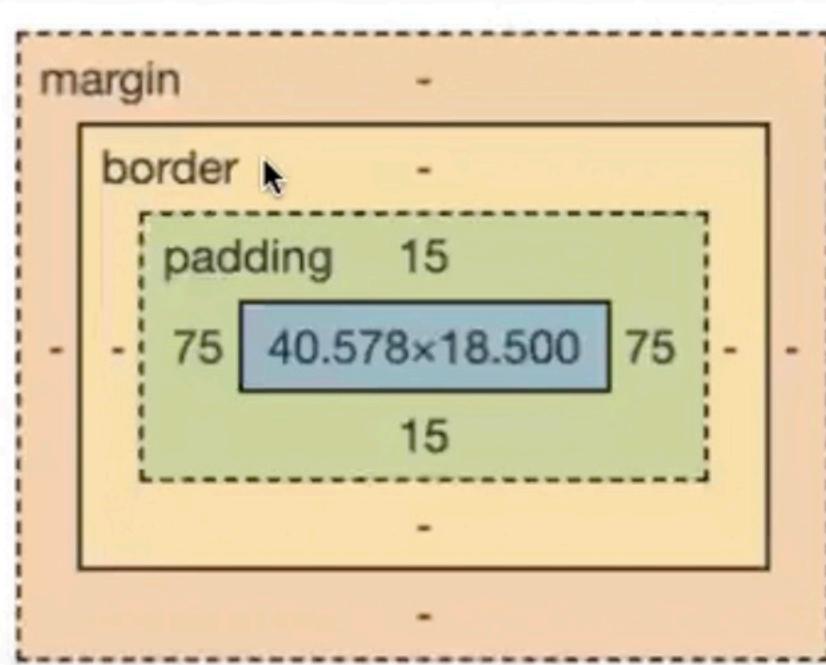
- HTML elemanlarının CSS özellikleri, etkilendikleri tüm stiller birleştirilerek oluşturulur.

## Specificity: Özgünlük

- Spesifik olma, genel olmaya göre daha baskındır.
- Inline > Id > Class > Tag**

# CSS Box Model

- Browser, her HTML elemanını bir kutu gibi görür.
- Tüm elemanların **content**, **padding**, **border** ve **margin** olmak üzere içten dışa 4 katmandan oluşur.



Element Structure:

```
::before
> <ul class="nav nav-tabs hidden-xs" id="crossDictionaryTabs">
  <div class="row tureng-searchresults-content" style="height: 100%; min-height: 0px !important;">
    <div class="col-lg-8 col-lg-offset-2 col-md-8 col-md-offset-0 tureng-searchresults-col-left" style="padding-top: 0px !important; height: 100%; min-height: 0px !important;">
      <div class="hidden-sm tureng-searchresults-ad-left" style="transform: translateX(0px); position: fixed; left: unset; right: 0px; margin-right: 350.5px; text-align: right; height: 0px;">_</div>
    </div>
  </div>
</ul>
```

Computed Tab:

Style Attribute:

```
E Style Attribute {
```

Media (min-width: 992px) and (max-width: 1199px)

- #crossDictionaryTabs {  
margin-left: 147px;}
- #crossDictionaryTabs {  
margin-left: -15px;  
margin-right: -15px;  
min-height: 40px;  
border-bottom: 0;}
- .nav-tabs {  
border-bottom: 3px solid #ccc;}
- .nav {  
padding-left: 0;  
margin-bottom: 0;  
list-style: none;}
- ul, ol {  
margin-top: 0;  
margin-bottom: 10px;}
- \* {  
:active  
:focus  
:focus-visible  
:focus-within  
:hover  
:target  
:visited}

Box Model:

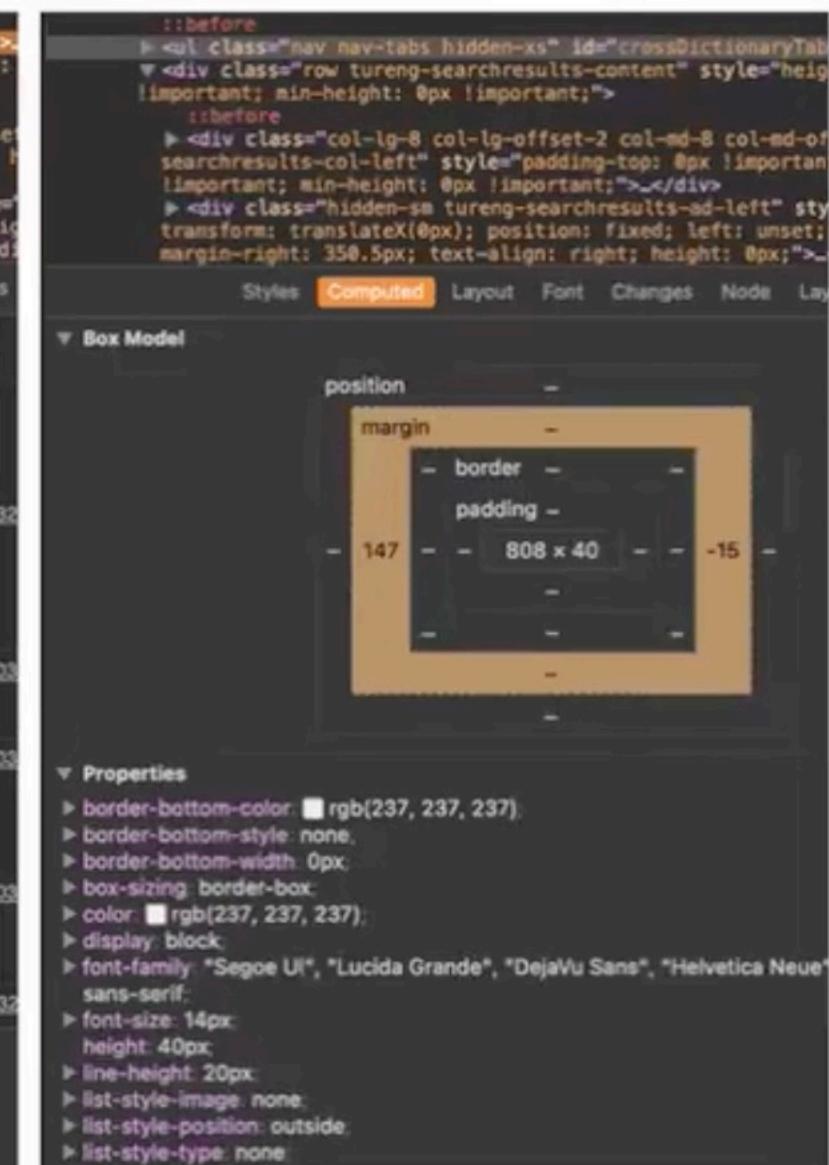
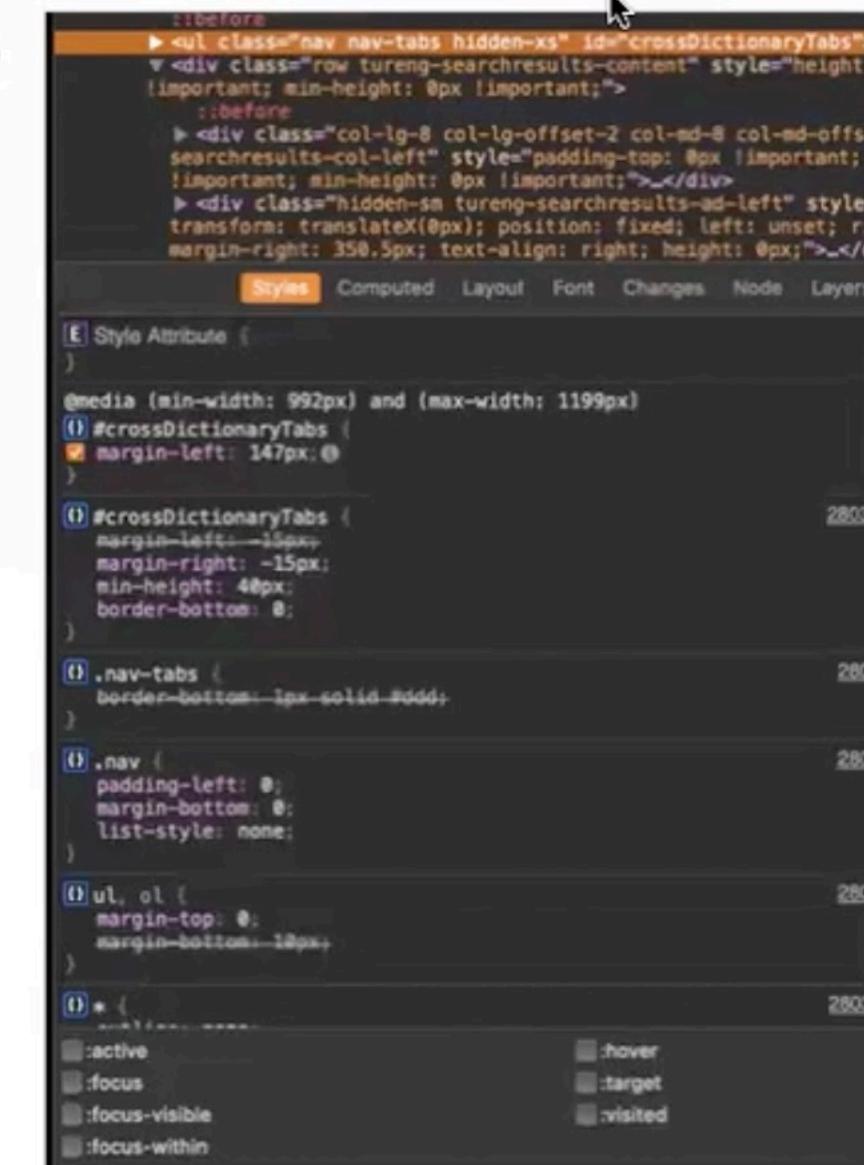
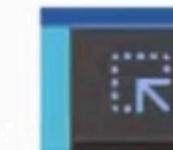
position	-
margin	-
border	-
padding	-
width	808
height	40
left	-147
top	-15

Properties:

- border-bottom-color: #rgb(237, 237, 237);
- border-bottom-style: none;
- border-bottom-width: 0px;
- box-sizing: border-box;
- color: #rgb(237, 237, 237);
- display: block;
- font-family: "Segoe UI", "Lucida Grande", "DejaVu Sans", "Helvetica Neue", sans-serif;
- font-size: 14px;
- height: 40px;
- line-height: 20px;
- list-style-image: none;
- list-style-position: outside;
- list-style-type: none;

# Dev Tools

- Sayfaya sağ tıklayarak, **inspect(incele)** seçeneği ile **dev tools**'u açabiliriz.
- Dev tools'da sol üstteki selector ile element seçebiliriz.
- Element'in etkilendiği stil bilgilerinin nerden geldiğini **Styles** sekmesinden ulaşabiliriz.
- Etkilendiği tüm style'ların bir araya gelip oluşturduğu final haline **Computed** sekmesinden ulaşabiliriz.
- Burada ayarları değiştirecek browser üzerinde yaptığımız ayarların sonuçlarını gözlemlayabiliriz.



The screenshot shows the Chrome DevTools interface with three main tabs visible: Elements, Styles, and Computed. The Elements tab shows the DOM tree with the 'crossDictionaryTabs' element selected. The Styles tab displays a list of CSS rules contributing to the selected element's style, including media queries and specific declarations like margin-left and border-bottom. The Computed tab shows the final, cumulative style properties for the element, including margin, padding, border, and font details. A large orange box highlights the 'margin-left: 147px' declaration in both the Styles and Computed tabs.

# HTML Gruplama Tag'leri

- Gruplama tag'leri ile birçok elemanı seçip onlara stil uygulayabiliriz.
- CSS ile sayfa düzeni -yani layout oluştururken, çok işimize yarar.
- En bilindik 2 tanesi:
  - **div**, block gruplamalar için kullanılır
  - **span**, inline elemanlar için kullanılır.

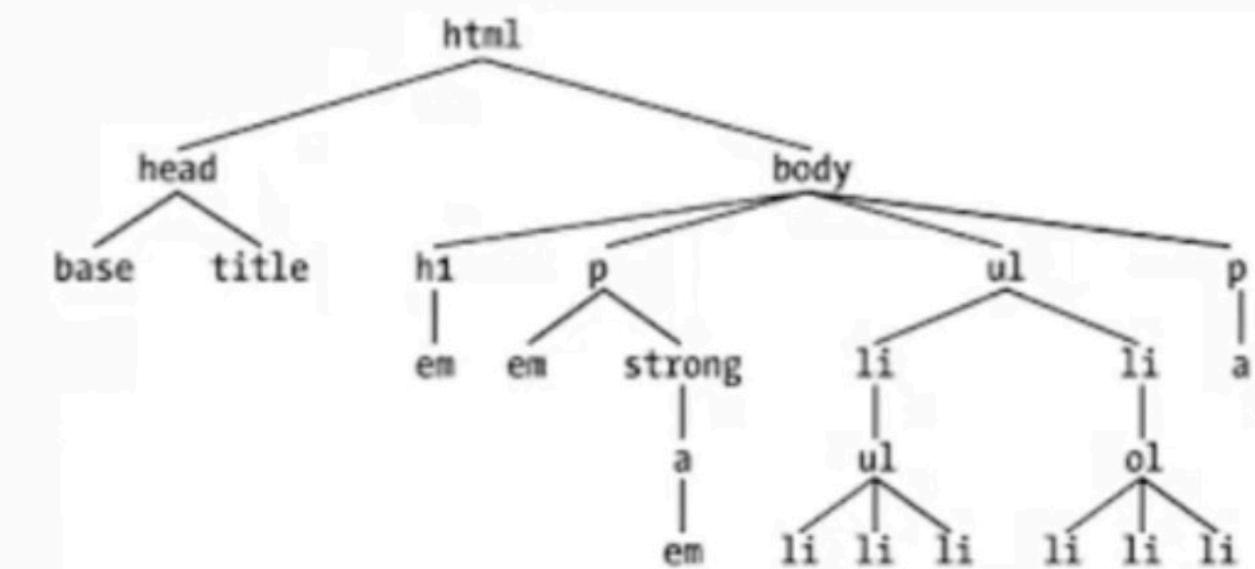
# İleri CSS Seçicileri: Combined selectors

- HTML kodunun yapısı bir aile ağaçına benzer.
- Hiyerarşik ilişkileri, bir elemanı seçtikten sonra başka bir elemana erişmek için kullanabiliriz.
  - Selectorleri boşluk kullanarak birleştirmek

```
body p { color: red; }
```

- Selectorleri peşpeşe yazarak birleştirmek

```
div.dark { color: red; }
```



# İleri CSS Seçicileri: Pseudo-class

Arada boşluk olmadan yazılır.

- **:hover**

```
.tweet-button:hover {  
    background-color: #198CD8;  
}
```

- **:first-child**

```
li:first-child {  
    color: orange;  
}
```

- **:last-child**

```
li:last-child {  
    border-bottom: 0;  
}
```

# Flex Container Properties

2 tane axis vardır. Default axis(direction ile aynı yön) ve cross-axis(direction'a dikey yön).

- **flex-direction**: itemların yatay veya dikey yerleştirir
- **justify-content**: default yönde itemların yerleşimini belirtir.
  - **space-between**: ilk item'ı başa, son item'ı sona dayar ve aradan kalan boşluğu eşit yapar.
  - **space-around**: her item'ın başında ve sonundaki boşluğu eşit yapar.
  - **space-evenly**: her boşluğu(başta, arada, sonda) eşit yapar.
- **align-items**: justify content gibi ama cross-axis'teki yerleşimi belirtir.
- **gap**: itemlar arasındaki boşluğu belirtir.
- **flex-wrap**: Sığmayan itemların yeni bir satır/sütun oluşturup(wrap) oluşturmayacağı(no-wrap) belirtir.



flex-direction

## Özellik Nedir?

`flex-direction` özelliği, CSS Flexbox düzeninde flex öğelerinin konteynere içinde hangi yönde sıralanacağını belirler. Bu özellik, flex konteynerine uygulanır ve içindeki flex öğelerinin yatay veya dikey olarak sıralanmasını sağlar.

flex-direction

## Değerleri

`flex-direction` özelliği aşağıdaki değerleri alabilir:

- `row` (**Varsayılan**): Flex öğelerini soldan sağa doğru yatay olarak sıralar.
- `row-reverse`: Flex öğelerini sağdan sola doğru yatay olarak sıralar.
- `column`: Flex öğelerini yukarıdan aşağıya doğru dikey olarak sıralar.
- `column-reverse`: Flex öğelerini aşağıdan yukarıya doğru dikey olarak sıralar.



`justify-content` özelliği, CSS Flexbox düzende flex öğelerinin ana eksen boyunca nasıl hizalanacağını belirler. Bu özellik, flex konteynerine uygulanır ve içindeki flex öğelerinin yatay veya dikey olarak nasıl hizalanacağını kontrol eder.

### `justify-content` Değerleri

`justify-content` özelliği aşağıdaki değerleri alabilir:

- `flex-start` (**Varsayılan**): Flex öğelerini ana eksenin başlangıcına (varsayılan olarak soldan sağa) hizalar.
- `flex-end`: Flex öğelerini ana eksenin sonuna hizalar.
- `center`: Flex öğelerini ana eksenin ortasına hizalar.
- `space-between`: Flex öğelerini konteyner içinde eşit aralıklarla dağıtır. İlk öğe konteynerin başlangıcına, son öğe ise konteynerin sonuna hizalanır.
- `space-around`: Flex öğelerini konteyner içinde eşit aralıklarla dağıtır. Her öğenin etrafında eşit boşluk bulunur.
- `space-evenly`: Flex öğelerini konteyner içinde eşit aralıklarla dağıtır. Her öğenin etrafında eşit boşluk bulunur.



`align-items` özelliği, CSS Flexbox düzeninde flex öğelerinin çapraz eksen boyunca nasıl hizalanacağını belirler. Bu özellik, flex konteynerine uygulanır ve içindeki flex öğelerinin dikey olarak nasıl hizalanacağını kontrol eder.

### `align-items` Değerleri

`align-items` özelliği aşağıdaki değerleri alabilir:

- `stretch` (**Varsayılan**): Flex öğelerini konteyner boyunca uzatır.
- `flex-start`: Flex öğelerini çapraz eksenin başlangıcına (varsayılan olarak yukarı) hizalar.
- `flex-end`: Flex öğelerini çapraz eksenin sonuna hizalar.
- `center`: Flex öğelerini çapraz eksenin ortasına hizalar.
- `baseline`: Flex öğelerini metin taban çizgilerine göre hizalar.

### Örnekler

Aşağıdaki örneklerle `align-items` özelliğinin nasıl çalıştığını daha iyi anlayabiliriz:



gap özelliği, CSS Flexbox ve Grid düzenlerinde öğeler arasındaki boşluğu ayarlamak için kullanılan bir özelliktir. Bu özellik, öğelerin arasındaki mesafeyi kolayca belirlemeyi sağlar.

### gap Özelliği Değerleri

gap özelliği, boşluk miktarını belirtmek için farklı değerler alabilir:

- **Tek bir değer:** Hem yatay hem de dikey boşluğu aynı anda ayarlar. Örneğin, gap: 10px; hem yatay hem de dikey boşluğu 10 piksel olarak ayarlar.
- **İki değer:** İlk değer yatay boşluğu, ikinci değer ise dikey boşluğu ayarlar. Örneğin, gap: 10px 20px; yatay boşluğu 10 piksel, dikey boşluğu ise 20 piksel olarak ayarlar.

### gap Özelliği Nasıl Kullanılır?

gap özelliği, flex konteynerine veya grid konteynerine uygulanır. Bu konteynerlerin içindeki öğeler arasındaki boşluğu ayarlamak için kullanılır.

## Örnekler

Aşağıdaki örneklerle gap özelliğinin nasıl çalıştığını daha iyi anlayabiliriz:



`flex-wrap` özelliği, CSS Flexbox düzeninde flex öğelerinin konteyner içinde sığmadığında nasıl davranışacağını belirler. Bu özellik, flex konteynerine uygulanır ve içindeki flex öğelerinin tek bir satırda mı kalacağını, yoksa alt satırlara kayıp kaymayacağını kontrol eder.

### `flex-wrap` Değerleri

`flex-wrap` özelliği aşağıdaki değerleri alabilir:

- `nowrap` (**Varsayılan**): Flex öğelerini tek bir satırda tutar. Öğeler konteyner içinde sığmadığında taşıma yaşanır.
- `wrap`: Flex öğelerini konteyner içinde sığmadığında alt satırlara kaydırır.
- `wrap-reverse`: Flex öğelerini konteyner içinde sığmadığında alt satırlara kaydırır, ancak sıralamayı tersine çevirir.

### `flex-wrap` Özelliği Nasıl Kullanılır?

`flex-wrap` özelliği, flex konteynerine uygulanır. Bu konteynerin içindeki öğelerin nasıl sarılacağını belirlemek için kullanılır.

### Örnekler

Aşağıdaki örneklerle `flex-wrap` özelliğinin nasıl çalıştığını daha iyi理解abiliriz:

# Flex Item Properties

- **flex-grow**: Item'ın genişleme katsayısıdır. Flex-containerda boş yer varsa hangi oranda genişleyebileceğini belirtir.
- **flex-basis**: Item'ların başlangıç büyüklüklerini belirtir. Default değeri autodur, yani flex-itemları kendi içerikleri kadar yer kaplar.
- **flex-shrink**: flex-grow'un tam tersidir: Eğer flex container, flex-itemlarının sığacağı kadar büyük değilse, elemanların ne kadar daralacağını belirler.

# Layout Oluşturmak

Best practices;

## 1. Yatay eksende içeriği ortalamak

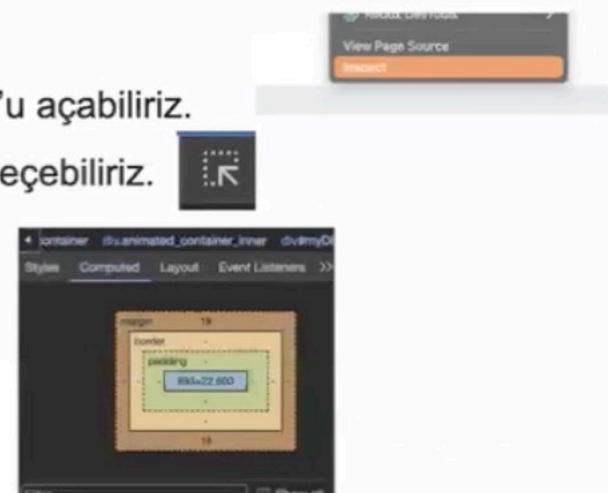
- Display property değerini block özellikle bir değer ver: Örn: **display: block;**
- width ayarla: Örn: **width: 650px;**
- margin-left ve margin-right değerini auto yap: Örn: **margin: 0 auto;**

## 2. Reset.css oluşturmak

- Ön tanımlı stiller browser'dan browser'a değişiklik gösterebilir.
- hazır reset.css kullanarak tüm stilleri kendimiz yazabiliriz.

## 3. Devtools ile stilleri öğrenip aktarmak

- Sayfamızda sağ tıklayıp inspect(ögeyi incele) seçeneği ile dev tools'u açabiliriz.
- Dev tools'da sol üst köşedeki selector ile sayfamızdan bir element seçebiliriz.
- Dev tools'da computed sekmesini açarak uygulana stilleri görebilir, üzerinde değişiklik yaparak etkisini deneyimleyebiliriz.
- Final css stillerini de kodumuza aktarabiliriz.

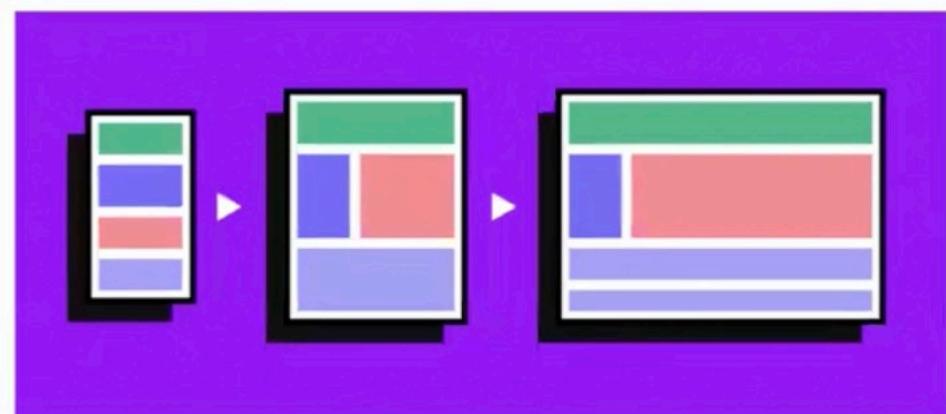


# Responsive Tasarım

Bir sitenin farklı cihaz ve ekran boyutlarında düzgün görüntülenmesine yönelik bir yaklaşımdır.

Kullanıcı deneyimini optimize etmek için web sayfalarını ve içerikleri otomatik olarak yeniden düzenlemek ve ölçeklemek için kullanılır.

- **Fixed:** Sabit ölçüler kullanılır: **px**
- **Fluid:** Oransal ölçüler kullanılır: **%, vw**
- **Adaptive:** Breakpointler kullanılır: **max-width: 420px**
- **Responsive:** Fluid + Adaptive birleşimidir.



[Güzel bir kaynak](#)

# Oransal Ölçüler

- **%** işaretini **parent'ın genişliğine** göre hesap yapar.
- **vw**, viewport width'i yani browserda sayfanın görüntülendiği kısmın **genişliğini** baz alır.
  - Sayfa genişliğinin tamamı 100vw'ye eşittir.
  - Bu durumda 1 vw, sayfa genişliğinin yüzde birine eşittir.
- **vh**, viewport height'i yani browserda sayfanın görüntülendiği kısmın **yüksekliğini** baz alır.
  - Sayfa yüksekliğinin tamamı 100vh'ye eşittir.
  - Bu durumda 1vh, sayfa yüksekliğinin yüzde birine eşittir.
- **rem**: html tag'inin font-size değeri tüm css dosyasında 1rem olarak kabul edilir.
  - Bu açıdan vw'ye benzer.
  - Bu sayede tüm CSS dosyasında font sizeler arasında oransal bir ilişki kurulabilir.
- **em**: Elemanların miras olarak aldığı font-size'ı 1em olarak kabul edilir.
  - Örneğin sayfamızda body tag'inin font-size'ı 20px olsun.
  - body tag'inin child'ı konumunda olan p'lere stil yazarken 1em değeri 20px'e eşit olur.
  - Parentına oranlı olması açısından '%'ye benzer.

# Media Query'leri

Media queryler sayesinde cihazların genişlik, yükseklik, yön, çözünürlük gibi özelliklerine bağlı olarak geçerli olacak CSS yazılabilir.

- Media queryler içinde kullanılan ve değişikliğin olacağı noktaları temsil eden değerlere **breakpoint** denir.
- Genişlik özelinde telefon, tablet, laptop ve büyük ekranlar için yaygın kullanılan değerler vardır.
- Media query'lerin CSS spesifisitesinden olumsuz etkilenmemesi için **dosyaların en sonuna yazılır.**

```
@media (max-width: 768px) {  
    body {  
        color: white;  
        background-color: black;  
    }  
}
```

telefonlar	w < 420px
büyük telefonlar ve küçük ekranlı tabletler	420px ≤ w < 768px
tabletler	768px ≤ w < 1024px
laptoplar	1024px ≤ w < 1280px
büyük ekranlar	1280px ≤ w

# Ekstra: Dev Tools

Dev Tools'da **Toggle Device Toolbar**  ikonuna tıklayarak farklı ekranlar için sayfamızı test edebiliriz.

