# Bootstrap Flexbox Nedir?

**Flexbox**, (ya da **Flexible Box Layout**), CSS ile esnek ve dinamik düzenler oluşturmak için kullanılan bir düzenleme modelidir. **Bootstrap**, 4. sürümünden itibaren **Flexbox** sistemini varsayılan olarak kullanmaya başlamıştır. Bu, sayfa elemanlarının yatayda veya dikeyde esnek bir şekilde hizalanmasını ve yerleşmesini sağlar. Flexbox, geleneksel CSS düzenleme yöntemlerine göre çok daha güçlü ve esnektir, çünkü öğeleri esnek bir şekilde düzenler ve farklı ekran boyutlarına göre uyum sağlar.

Bootstrap'ta **Flexbox**, elemanları hizalamak, sıralamak, büyütmek veya küçültmek için çeşitli yardımcı sınıflar (utility classes) sunar. Bu sayede sayfa düzenlerinde, özellikle de duyarlı (responsive) tasarımlarda büyük kolaylık sağlar.

# Flexbox ile Kullanılabilen Temel Özellikler

## 1. Flex Container (Esnek Kapsayıcı)

Flexbox ile çalışmaya başlamak için bir **flex container** (esnek kapsayıcı) oluşturmanız gerekir. Flex container, içinde bulunan öğelere **flex item** (esnek öğe) özelliklerini uygulayacaktır.

`.d-flex`: Bir öğeyi flex container yapar.

`.d-inline-flex`: Flex container yapar ama öğenin satır içi (inline) olmasını sağlar.

### Örnek:

```
<div class="d-flex">
 <!-- Flex öğeler burada olacak -->
</div>
```

## 2. Flex Items (Esnek Öğeler)

Flex container içerisine eklediğiniz her öğe, otomatik olarak **flex item** haline gelir. Flex item'lar, kapsayıcıları içinde esnek bir şekilde sıralanabilir ve hizalanabilir.

### Flexbox ile Kullanılabilen Temel Özellikler

## 1.Flexbox Hizalama (Alignment) Nedir?

**Flexbox hizalama**, CSS Flexbox düzeninde öğelerin yerleşimini ve hizalanmasını kontrol etmek için kullanılan özelliklerin tümüdür. Flexbox, öğeleri **horizontally** (yatayda) ve **vertically** (dikeyde) hizalamayı kolaylaştıran bir sistemdir. Flexbox düzeni, özellikle responsive (duyarlı) web tasarımında, öğeleri farklı ekran boyutlarına göre esnek bir şekilde hizalamayı sağlar.

Flexbox'ta hizalama, **container** (kapsayıcı) ve **item** (öğe) düzeyinde yapılabilir. Yani hem kapsayıcının içindeki öğeleri hizalayabiliriz, hem de öğenin kendi içinde (diğer öğelere göre) nasıl hizalanacağını belirleyebiliriz.

## 1.Flexbox Hizalama Özellikleri

### 1. Kapsayıcıda (Container) Hizalama

Kapsayıcıda yapılan hizalama, **flex container** (flex kapsayıcı) içindeki öğelerin nasıl sıralanacağı, hizalanacağı ve dağıtılacağı ile ilgilidir. Bunun için kullanılan temel özellikler şunlardır:

a) `justify-content` (Yatay Hizalama)

Bu özellik, öğeleri **ana eksende** (main axis) hizalar. Ana eksen, genellikle yatay (soldan sağa) olur, ancak `flex-direction` özelliğine bağlı olarak dikey de olabilir.

`justify-content: flex-start`: Öğeleri **başlangıç noktasına** (sola) hizalar.

`justify-content: flex-end`: Öğeleri **son noktasına** (sağa) hizalar.

`justify-content: center`: Öğeleri **ortalar** (yatayda).

`justify-content: space-between`: Öğeler arasına **eşit boşluk** bırakır, ilk öğe başlangıç noktasına, son öğe ise son noktaya yerleşir.

`justify-content: space-around`: Öğeler arasında **eşit boşluk** bırakır, ancak kenarlarda da boşluk olur.

`justify-content: space-evenly`: Öğeler arasında **eşit boşluk** bırakır ve kenar boşlukları da eşittir.

Örnek:

```
<div style="display: flex; justify-content: center;">
 <div>Öğe 1</div>
 <div>Öğe 2</div>
 <div>Öğe 3</div>
</div>
```

Bu örnekte, öğeler **yatayda ortalanır**.

b) `align-items` (Dikey Hizalama)

Bu özellik, öğeleri **çapraz eksende** (cross axis) hizalar. Çapraz eksen, genellikle dikey (yukarıdan aşağıya) olur, ancak `flex-direction` özelliğine bağlı olarak yatay da olabilir.

`align-items: flex-start`: Öğeleri **üstte** hizalar.

`align-items: flex-end`: Öğeleri **altta** hizalar.

`align-items: center`: Öğeleri **dikeyde ortalar**.

`align-items: baseline`: Öğeleri, metinlerinin **taban çizgisine göre** hizalar.

`align-items: stretch`: Öğeleri, kapsayıcının **yüksekliğine kadar** uzatır (varsayılan).

Örnek:

```
<div style="display: flex; align-items: center; height: 100px;">
 <div>Öğe 1</div>
 <div>Öğe 2</div>
 <div>Öğe 3</div>
</div>
```

Bu örnekte, öğeler **dikeyde ortalanır**.

## 2. Öğe İçinde (Item) Hizalama

Öğeler, kendi içinde hizalanabilir. Flexbox ile, her bir öğenin **içerik hizalamasını** kontrol edebilirsiniz.

a) `align-self` (Özelleştirilmiş Hizalama)

Bu özellik, **tekil bir öğeyi** diğer öğelerden bağımsız olarak hizalamaya olanak tanır. Örneğin, tüm öğeler `align-items: center` ile ortalanmışsa, bir öğe `align-self` ile farklı bir şekilde hizalanabilir.

`align-self: auto`: Varsayılan davranış (flex container'daki `align-items` ile hizalanır).

`align-self: flex-start`: Öğeyi **başlangıç noktasına** hizalar.

`align-self: flex-end`: Öğeyi **son noktasına** hizalar.

`align-self: center`: Öğeyi **ortalar** (dikeyde).

`align-self: baseline`: Öğeyi metin taban çizgisine göre hizalar.

`align-self: stretch`: Öğeyi, kapsayıcının yüksekliğine kadar uzatır.

Örnek:

```
<div style="display: flex; align-items: center; height: 100px;">
  <div>Öğe 1</div>
  <div style="align-self: flex-start;">Öğe 2</div>
  <div>Öğe 3</div>
</div>,
```

Bu örnekte, **Öğe 2** diğer öğelerden farklı olarak **üstte hizalanır**.

## `flex-start`, `center`, `flex-end` ve Diğer Değerlerin Anlamı

**`flex-start`**: Öğeleri **başlangıç noktasına** hizalar. Eğer **`flex-direction`** yataysa, soldan; dikeyse, yukarıdan hizalanır.

**`flex-end`**: Öğeleri **son noktaya** hizalar. Yatayda sağa, dikeyde ise alta hizalar.

**`center`**: Öğeleri **tam ortada hizalar**, hem yatay hem dikeyde.

**`baseline`**: Öğeleri **metin taban çizgisine göre hizalar**. Özellikle metin içeren öğelerde faydalıdır.

**`stretch`**: Öğeleri **kapsayıcının boyutuna kadar** uzatır (varsayılan değeri genellikle yükseklik için geçerlidir).

## Özet:

Flexbox hizalama, öğeleri hem **yatayda** hem de **dikeyde** hizalamayı kolaylaştıran güçlü bir sistem sunar. İki ana hizalama düzeyi vardır:

**Kapsayıcıda hizalama** (örn. `justify-content`, `align-items`) — Flex container içindeki tüm öğeleri hizalar.

**Öğe içindeki hizalama** (örn. `align-self`) — Tekil öğe üzerinde özelleştirilmiş hizalama yapar.

Bu özellikler sayesinde, Flexbox ile karmaşık yerleşimler oluşturmak çok daha kolay hale gelir ve öğelerin farklı ekran boyutlarına duyarlı bir şekilde hizalanması sağlanabilir.

### 2. Flex Direction (Esnek Yön)

**`flex-direction`** özelliği, öğelerin hangi eksende sıralanacağını belirler. Bootstrap'ta, bu özellik **`.flex-row`** ve **`.flex-column`** sınıflarıyla uygulanır.

**`.flex-row`**: Öğeleri **yatay** (soldan sağa) olarak sıralar.

**`.flex-column`**: Öğeleri **dikey** (yukarıdan aşağıya) olarak sıralar.

**`.flex-row-reverse`**: Öğeleri **ters yönde** yatay olarak sıralar (sağdan sola).

**`.flex-column-reverse`**: Öğeleri **ters yönde** dikey olarak sıralar (aşağıdan yukarıya).

**Örnek:**

```
<div class="d-flex flex-column">
  <div>Öğe 1</div>
```

```
  <div>Öğe 2</div>
</div>
```

Bu örnekte, öğeler **dikey olarak sıralanır**.

## 3. Flex-wrap Nedir?

**`flex-wrap`**, CSS Flexbox düzeninde kullanılan bir özelliktir ve **flex öğelerinin** (flex items) konteynerin sınırlarını aştığı durumlarda nasıl davranacaklarını belirler. Özellikle **çok sayıda öğenin bulunduğu** veya **flex öğelerinin genişliğinin** container'dan fazla olduğu durumlarda kullanılır. Bu özellik, öğelerin **sarılmasını** (wrap) ya da **tek satırda** kalmasını kontrol eder.

## Temel Kavram

Flexbox varsayılan olarak, flex öğelerini tek bir satıra yerleştirir ve öğeler yeterli alana sığmazsa, **overflow** (taşma) durumuna yol açar. **`flex-wrap`** özelliği, öğelerin taşma yerine **sarılmasını** sağlayarak, yeni bir satırda ya da sütunda hizalanmalarını sağlar.

## `flex-wrap` Değerleri

**`nowrap`** (varsayılan): Bu değer, öğelerin **tek bir satıra** yerleştirilmesini sağlar. Öğeler, konteynerin genişliğine sığmasa bile taşmaya (overflow) devam eder.

**`wrap`**: Bu değer, öğelerin **sarılmasına** izin verir. Yani, öğeler konteynerin boyutlarından fazla yer kapladığında, fazla olan öğeler bir sonraki satıra (veya sütuna) geçer.

**`wrap-reverse`**: Bu değer de öğelerin **sarılmasını** sağlar, ancak öğeler **ters yönde** sarılır. Yani, öğeler normalde yukarıdan aşağıya doğru sarılacakken, bu seçenekle öğeler aşağıdan yukarıya doğru sarılır.

## Kullanım Örnekleri

### 1. `nowrap` (Varsayılan Davranış)

### Örnek:

```
<div style="display: flex; flex-wrap: nowrap;">
  <div style="width: 100px; height: 100px; background-color: red;">Öğe 1</div>
  <div style="width: 100px; height: 100px; background-color: green;">Öğe 2</div>
  <div style="width: 100px; height: 100px; background-color: blue;">Öğe 3</div>
</div>
```

Bu örnekte, `flex-wrap: nowrap` olduğu için öğeler tek bir satırda yer alır. Eğer öğeler konteynerin genişliğinden fazla yer kaplarsa, taşma (overflow) olur.

### 2. `wrap` (Sarılma)

```
<div style="display: flex; flex-wrap: wrap; width: 250px;">
  <div style="width: 100px; height: 100px; background-color: red;">Öğe 1</div>
  <div style="width: 100px; height: 100px; background-color: green;">Öğe 2</div>
  <div style="width: 100px; height: 100px; background-color: blue;">Öğe 3</div>
</div>
```

### 3. `wrap-reverse` (Ters Sarılma)

```
<div style="display: flex; flex-wrap: wrap-reverse; width: 250px;">
  <div style="width: 100px; height: 100px; background-color: red;">Öğe 1</div>
  <div style="width: 100px; height: 100px; background-color: green;">Öğe 2</div>
  <div style="width: 100px; height: 100px; background-color: blue;">Öğe 3</div>
</div>
```

Bu örnekte, `flex-wrap: wrap-reverse` kullanıldığı için, öğeler ters sırayla sarılır. Bu durumda **Öğe 3** en üstte, **Öğe 1** en altta olacak şekilde sarılacaktır.

## `flex-wrap` Kullanım Senaryoları

**Responsive Design**: `flex-wrap`, duyarlı (responsive) tasarımlar oluştururken özellikle faydalıdır. Ekran boyutu küçüldüğünde öğeler **sarılabilir**, böylece daha küçük ekranlarda öğeler düzgün bir şekilde sıralanır.

**Dinamik İçerik**: Eğer öğelerin boyutları dinamikse veya öğe sayısı bilinmiyorsa, `flex-wrap` öğelerin birbirine girmesini engeller.

**Grid Layouts**: `flex-wrap`, basit grid (ızgara) yapıları oluşturmak için de kullanılabilir. Öğeler sarıldığında, yeni satırlarda düzenli bir şekilde yerleşirler.

## Özet

`flex-wrap`, Flexbox ile öğelerinizin konteynerde **sarılmasını** kontrol eder. Bu özellik, öğeler fazla yer kapladığında öğelerin **sarılmasına** veya **taşmasına** (overflow) olanak tanır.

`nowrap`: Öğeler tek satıra yerleştirilir (varsayılan).

`wrap`: Öğeler sarılır ve yeni satırlara geçer.

`wrap-reverse`: Öğeler ters sırayla sarılır.

`flex-wrap` ile, esnek ve duyarlı (responsive) düzenler yaratabilir ve öğelerinizi farklı ekran boyutlarına göre düzgün bir şekilde hizalayabilirsiniz.

### 4. Order (Sıra)

Flexbox ile öğelerin sırasını değiştirebilirsiniz. Bootstrap'ta, öğelerin sırasını değiştirmek için `order` sınıflarını kullanabilirsiniz.

`.order-1`, `.order-2`, `.order-3`,...: Öğelere sırasıyla farklı numaralar atar.

**Örnek:**

```
<div class="d-flex">
  <div class="order-3">Öğe 1</div>
  <div class="order-1">Öğe 2</div>
  <div class="order-2">Öğe 3</div>
</div>
```

Bu örnekte, öğeler **sırasıyla değiştirilir**.

## `5.align-self` (Özelleştirilmiş Hizalama)

`align-self`, Flexbox düzeninde kullanılan bir CSS özelliğidir ve **tekil bir flex öğesinin** (flex item) kendi içinde **dikeyde** (veya çapraz eksende) nasıl hizalanacağını belirler. Bu özellik, **flex container**

(flex kapsayıcı) üzerindeki `align-items` özelliğine göre **bireysel öğelerin hizalanmasını özelleştirmeye** olanak tanır.

## Temel Kavramlar

`align-items`: Flex container içindeki **tüm öğelerin** hizalanmasını belirler. Örneğin, tüm öğeleri yukarıya, ortalamaya veya alta hizalayabilirsiniz.

`align-self`: **Tek bir öğeyi** özel olarak hizalamak için kullanılır. Yani, `align-items` ile belirlediğiniz genel hizalamayı **geçersiz kılmak** için kullanılır ve sadece hedef öğe üzerinde etkili olur.

## `align-self` Değerleri

`align-self` özelliği, flex item'lar (flex öğeleri) üzerinde aşağıdaki değerlerle kullanılabilir:

`auto`: Varsayılan değerdir. Bu değer, öğenin `align-items` özelliğine bağlı olarak hizalanmasını sağlar. Yani, container'daki genel hizalama nasıl ise, öğe o şekilde hizalanır

`flex-start`: Öğeyi **üstte** hizalar (başlangıç noktasına).

`flex-end`: Öğeyi **altta** hizalar (son noktaya).

`center`: Öğeyi **dikeyde ortalar**.

**`baseline`**: Öğeyi, öğenin metin taban çizgisine göre hizalar. Bu, özellikle metin içeren öğelerde faydalıdır.

**`stretch`**: Öğeyi, container'ın **yüksekliğine kadar uzatır** (stretch), yani öğenin yüksekliğini container'ın yüksekliğiyle aynı yapar.

## Kullanım Örneği

```
<div class="d-flex" style="height: 100px;">
  <div style="align-self: flex-start; background-color: red;">Öğe
1</div>
  <div style="align-self: center; background-color: green;">Öğe 2</div>
  <div style="align-self: flex-end; background-color: blue;">Öğe
3</div>
</div>
```

Bu örnekte:

**Öğe 1** (`align-self: flex-start`): Üstte hizalanır.

**Öğe 2** (`align-self: center`): Dikeyde ortalanır.

**Öğe 3** (`align-self: flex-end`): Altta hizalanır.

## `align-self` ile `align-items` Arasındaki Fark

**`align-items`**, **tüm öğelerin hizalanmasını** belirler.

**`align-self`**, sadece **tek bir öğe üzerinde hizalama yapmanızı** sağlar.

Örneğin, bir container’da tüm öğeleri ortalamak istiyorsanız, `align-items: center;` kullanabilirsiniz. Ancak sadece bir öğeyi ortalamak isterseniz, o öğeye **`align-self: center;`** uygulayabilirsiniz.

## Özet

**`align-self`**, Flexbox düzeninde, bir öğenin dikeyde (cross axis) hizalanmasını özelleştiren güçlü bir özelliktir. Bu özellik, öğelerin tek tek hizalanmasını kontrol etmenizi sağlar, böylece container içindeki diğer öğelerin hizalamasına etki etmeden belirli bir öğeyi istediğiniz gibi hizalayabilirsiniz. Bu, özellikle karmaşık layout'lar ve dinamik içerikler için çok faydalıdır.

`6. flex-shrink`

**`flex-shrink`** CSS'de, **Flexbox** düzeniyle kullanılan bir özelliktir ve bir **flex öğesinin** konteynerin boyutlarından daha küçük bir alana sığması gerektiğinde ne kadar küçülmesi gerektiğini belirler. Yani, **`flex-shrink`**, flex öğelerinin yerleştirileceği alan yetersiz olduğunda, öğelerin **ne kadar küçüleceğini** kontrol eder.

## `flex-shrink` Nasıl Çalışır?

Her bir flex öğesinin bir **küçülme faktörü** (shrink factor) vardır ve bu değer, öğenin diğer öğelere göre ne kadar küçülmesi gerektiğini belirler.

**`flex-shrink: 0`**: Bu durumda öğe **hiç küçülmez**. Eğer konteynerin boyutu yetersizse, öğe mevcut boyutunda kalır ve diğer öğelere yer bırakabilmek için alanı paylaşmaz.

**`flex-shrink: 1`** (varsayılan): Öğenin **normal şekilde küçülmesini** sağlar. Eğer diğer öğeler de aynı değere sahipse (yani `flex-shrink: 1` ise), öğeler birbirine orantılı bir şekilde küçülür.

**`flex-shrink: 2`**: Öğenin, **diğer öğelere göre iki kat daha fazla küçülmesini** sağlar.

## Örnek:

Aşağıdaki örnekte, üç öğenin farklı `flex-shrink` değerleri olduğunu görebilirsiniz.

```
<div style="display: flex; width: 400px;">
  <div style="flex-shrink: 1; width: 300px; background-color: red;">Öğe 1</div>
  <div style="flex-shrink: 2; width: 300px; background-color: green;">Öğe 2</div>
  <div style="flex-shrink: 0; width: 300px; background-color:
```

```
blue;">Öğe 3</div>
</div>
```

Bu örnekte:

**Öğe 1** (`flex-shrink: 1`): Eğer konteyner küçükse, öğe normal oranda küçülür.

**Öğe 2** (`flex-shrink: 2`): Bu öğe, diğer öğelere göre **iki kat daha fazla küçülür**.

**Öğe 3** (`flex-shrink: 0`): Bu öğe **hiç küçülmez** ve boyutunu korur.

## Sonuç:

**`flex-shrink`**, öğelerin daralan alana nasıl tepki vereceğini belirler. Eğer bir öğenin **küçülmesini istemiyorsanız**, `flex-shrink: 0` kullanabilirsiniz. Eğer öğenin **diğer öğelerden daha fazla küçülmesini** isterseniz, değeri artırarak kullanabilirsiniz.

## Özet

Bootstrap'ta **Flexbox** kullanımı, özellikle **esnek, duyarlı ve dinamik** düzenler oluşturmak için çok faydalıdır. **Yatay ve dikey hizalama**, **sıra değiştirme**, **sarılma (wrap)** gibi özelliklerle

elemanlarınızı rahatça düzenleyebilir, ekran boyutlarına göre uyumlu ve esnek tasarımlar oluşturabilirsiniz. Bootstrap, bu işlemleri kolaylaştırmak için çeşitli **yardımcı sınıflar (utility classes)** sunar ve böylece CSS yazmaya gerek kalmadan düzeninizi oluşturabilirsiniz.