WEB & YAZILIM GELİŞTİRME SERİSİ · MODÜL 4

CSS

Web'e görünüm: seçiciler, renkler ve tipografi; kutu modeli, Flexbox ve Grid ile düzen; responsive tasarım, geçişler ve animasyonlar. Gerçek kod ve özgün diyagramlarla, sıfırdan.

Kutu modeli · Flexbox · Grid · Responsive · Eğitim amaçlıdır

Bu Kitap Hakkında

Bu modül, HTML iskeletine görünüm kazandıran CSS'i sıfırdan öğretir. Dört seviye ve yirmi altı bölüm boyunca seçicilerden kutu modeline, Flexbox ve Grid ile düzen kurmaktan responsive tasarıma, geçiş ve animasyonlardan CSS değişkenlerine kadar her şeyi gerçek kod örnekleriyle ele alır.

Her bölümde çalışan kod blokları, konuyu görselleştiren özgün bir diyagram (kutu modeli, Flexbox eksenleri, Grid, responsive cihazlar), 'stil uygulanınca ne olur?' kartı ve bir alıştırma yer alır. Bu, on altı modüllük 'Web & Yazılım Geliştirme' serisinin dördüncü modülüdür; HTML ile kurduğun iskelete burada görünüm kazandırır, sıradaki modülde (JavaScript) etkileşim eklersin. Kod örneklerini kendi editöründe deneyerek ilerlemen önerilir. Bu seri eğitim amaçlıdır.

Web & Yazılım Geliştirme Serisi · Modül 4

İçindekiler

★ CSS Özellik Sözlüğü 61

SEVİYE 1

Renk ve Yazı

CSS'in temelleri: CSS nedir, CSS'i bağlamak, seçiciler, renkler ve arka plan, yazı tipi ve metin, birimler, kutu modeli ve temel görünüm (kenarlık, köşe, gölge).

BÖLÜM 01

CSS Nedir?

HTML sayfanın iskeletini kurar; ama renk, yerleşim, yazı tipi gibi görünümü CSS (Cascading Style Sheets) verir. CSS olmadan sayfalar düz ve biçimsizdir; CSS ile hayat bulur.

CSS ne yapar?

- Renkleri, arka planları, yazı tiplerini ayarlar.
- Öğeleri konumlandırır ve düzen (layout) kurar.
- Sayfayı farklı ekranlara uyarlar (responsive).

Şema 1.1 — CSS aynı HTML içeriğine görünüm kazandırır.

Nasıl çalışır?

- Bir **seçici** ile hangi öğeyi hedeflediğini söylersin.
- Bir **özellik** ve **değer** ile ona ne yapacağını belirtirsin.
- Örn: "tüm paragraflar mavi olsun".

İlk CSS kuralı

```
p {
  color: #2563EB;
  font-size: 18px;
}
```

İPUCU

HTML ile CSS'i ayrı tut: HTML "ne" (içerik), CSS "nasıl görünür" (stil) sorusuna cevap verir. Bu ayrım, kodu düzenli ve bakımı kolay tutar. Aynı CSS'i birçok sayfada kullanarak tutarlı bir görünüm sağlarsın.

Stil uygulanınca ne olur?

ETKİ

Yukarıdaki kural uygulanınca sayfadaki tüm paragraflar mavi renkte ve 18 piksel boyutunda görünür. İçerik (metin) aynı kalır; yalnızca görünümü değişir.

Alıştırma

8 dk

CSS'i tanı:

1. CSS'in yaptığı 3 şeyi kendi cümlelerinle yaz.
2. Bir CSS kuralının parçalarını (seçici, özellik, değer) say.
3. Neden HTML ve CSS ayrı tutulur, açıkla.

BÖLÜM 02

CSS'i Bağlamak

CSS'i HTML'e üç yolla ekleyebilirsin; ama profesyoneller neredeyse her zaman birini tercih eder. Hangisini, neden?

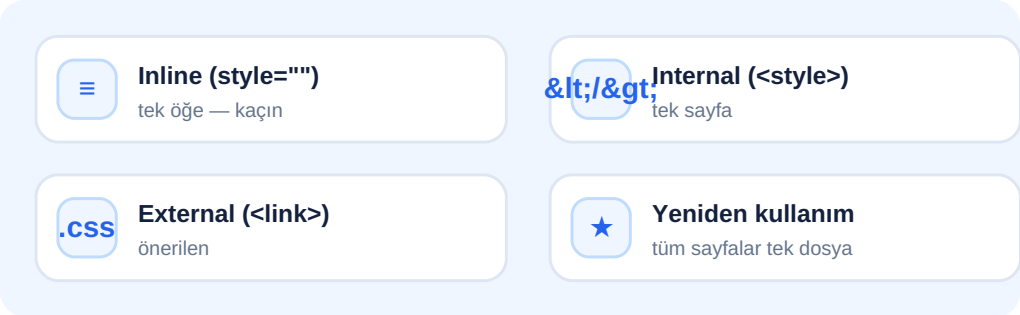
Üç yöntem

- **Satır içi (inline):** `style="..."` doğrudan etikete — küçük, dağınık.
- **İç (internal):** `<style>` bloğu `<head>` içinde — tek sayfalık.
- **Dış (external):** ayrı bir `.css` dosyası + `<link>` — en iyisi.

Dış CSS (önerilen yöntem)

```
<!-- index.html, head içinde -->
<link rel="stylesheet" href="styles.css">

/* styles.css */
body { font-family: sans-serif; }
```



Şema 2.1 — CSS'i bağlama yöntemleri; dış dosya en iyisidir.

İPUCU

Neredeyse her zaman **dış CSS dosyası** kullan: stillerin tek yerde toplanır, tüm sayfalara aynı dosyayı bağlarsın ve bir değişiklik her yere yansır. Inline stiller hızlı görünür ama kodu dağıtır ve bakımı zorlaştırır.

Stil uygulanınca ne olur? ETKİ

Dış CSS bağlanınca, `styles.css` 'teki tüm kurallar sayfaya uygulanır. Aynı dosyayı 10 sayfaya bağlarsan, hepsi tek tip görünür ve tek değişiklikle hepsini güncellersin.

Alıştırma

8 dk

CSS'i bağla:

1. Bir `styles.css` dosyası oluştur.
2. Onu `<link>` ile HTML'e bağla.
3. Bir kural ekleyip sayfada etkisini gör.

BÖLÜM 03

Seçiciler (Selectors)

Seçiciler, CSS'in hangi öğeleri etkileyeceğini söyler. Doğru seçiciyi seçmek, CSS'in kalbidir.

Bir CSS kuralının anatomisi

Şema 3.1 — Bir CSS kuralının parçaları: seçici, özellik, değer.

Sık kullanılan seçiciler

- **Element:** `p` — tüm paragraflar.
- **Sınıf (class):** `.uyari` — `class="uyari"` olan öğeler.
- **Kimlik (id):** `#baslik` — `id="baslik"` olan tek öğe.
- **Gruplama:** `h1, h2` — birden çok seçiciye aynı stil.

Farklı seçiciler

```
p { color: #333; }
.uyari { color: red; }
#baslik { font-size: 32px; }
h1, h2 { font-family: Georgia; }
```

İPUCU

En çok **sınıf (class) seçicileri** kullanırsın; esnektir ve birçok öğeye uygulanabilir. `id` seçicileri sayfada tek bir öğe içindir ve daha "güçlüdür" (özgüllük) — bunu Seviye 4'te göreceksin. Element seçicileri ise genel kurallar için iyidir.

Stil uygulanınca ne olur? ETKİ

`.uyari` kuralı, yalnızca `class="uyari"` taşıyan öğeleri kırmızı yapar; diğer öğeler etkilenmez. Doğru seçici, tam istediğin öğeleri hedeflemeni sağlar.

Alıştırma

10 dk

Seçicileri dene:

1. Bir element, bir class ve bir id seçici yaz.
2. Bir class'ı birden çok öğeye uygula.
3. İki seçiciyi gruplayıp (`,`) aynı stili ver.

BÖLÜM 04

Renkler ve Arka Plan

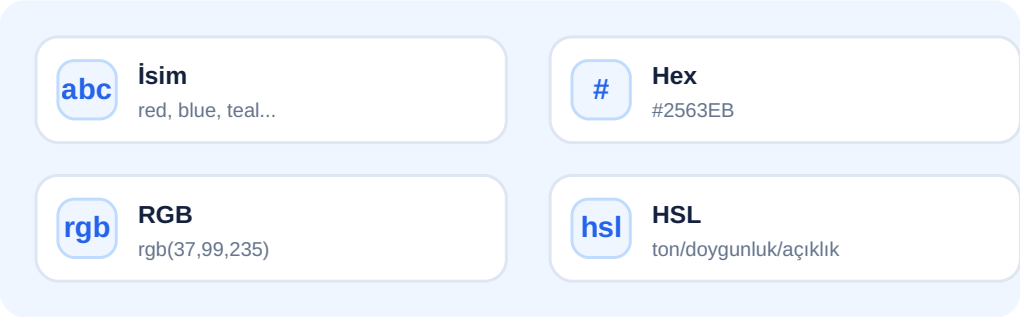
Renk, bir tasarımın ruhudur. CSS'te rengi birkaç farklı biçimde ifade edebilir, metin ve arka plan renklerini ayrı ayrı ayarlayabilirsin.

Renk ifade biçimleri

- **İsimle:** `red`, `blue` (sınırlı sayıda).
- **Hex:** `#2563EB` (en yaygın).
- **RGB:** `rgb(37, 99, 235)`.
- **HSL:** `hsl(221, 83%, 53%)` (ton-doygunluk-açıklık).

Renk ve arka plan

```
.kart {
  color: #1e293b;
  background-color: #EFF6FF;
}
```



Şema 4.1 — Aynı rengi farklı biçimlerde yazabilirsin.

İPUCU

Pratikte en çok **hex** (`#2563EB`) ve giderek **HSL** kullanılır; HSL, bir rengin tonunu koruyup açıklığını ayarlamayı kolaylaştırır. Renk seçerken kontrast'a dikkat et: metin ile arka plan arasında yeterli fark olmalı (okunabilirlik).

Stil uygulanınca ne olur? ETKİ

`color` metnin rengini, `background-color` ise arka plan rengini belirler. Yukarıdaki kart koyu metin + açık mavi arka planla görünür.

Alıştırma

8 dk

Renklerle oyna:

1. Bir öğeye `color` ve `background-color` ver.
2. Aynı rengi hex ve rgb ile yaz.
3. Metin/arka plan kontrastının okunur olduğunu kontrol et.

BÖLÜM 05

Yazı Tipi ve Metin

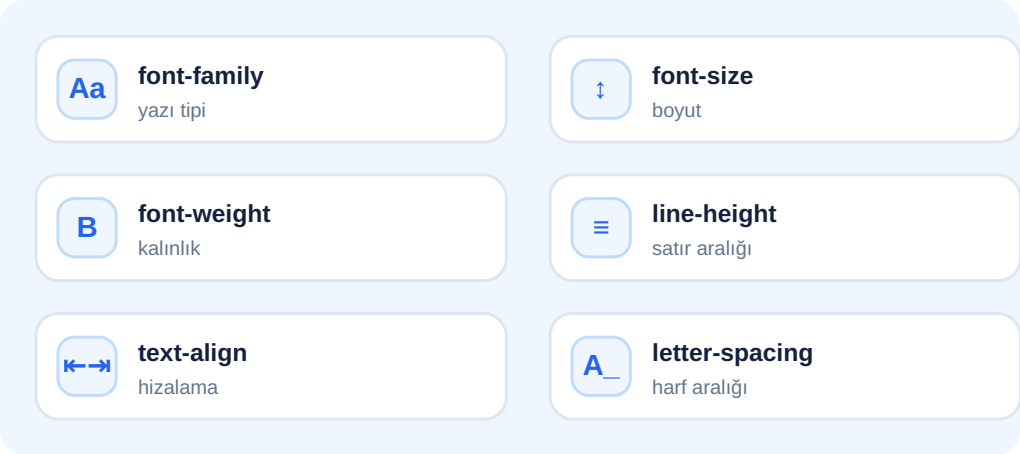
Tipografi (yazı düzeni), bir sayfanın okunabilirliğini ve karakterini belirler. CSS, yazı tipini ve metnin görünümünü ayrıntılı kontrol etmeni sağlar.

Temel metin özellikleri

- `font-family` — yazı tipi (örn. Arial, Georgia).
- `font-size` — boyut. `font-weight` — kalınlık.
- `line-height` — satır aralığı (okunabilirlik için önemli).
- `text-align` — hizalama (left, center, right).

Metin biçimlendirme

```
body {
  font-family: system-ui, sans-serif;
  font-size: 16px;
  line-height: 1.6;
}
h1 { font-weight: 700; text-align: center; }
```



Şema 5.1 — Temel tipografi özellikleri.

İPUCU

Okunabilirlik için **line-height**'ı ihmal etme; 1.5–1.7 arası satır aralığı metni çok daha rahat okunur kılar. Gövde metni için 16px ve üstü bir boyut tercih et. Sistem yazı tipleri (`system-ui`) hızlı yüklenir ve her cihazda doğal görünür.

Stil uygulanınca ne olur? ETKİ

Bu kurallar uygulanınca sayfa metni seçilen yazı tipinde, rahat satır aralığıyla görünür; `h1` kalın ve ortalanmış olur. Tipografi, içeriği değiştirmeden okuma deneyimini iyileştirir.

Alıştırma 8 dk

Tipografiyi ayarla:

1. Sayfaya bir `font-family` ve `line-height` ver.
2. Bir başlığı ortala ve kalınlaştır.
3. Satır aralığını değiştirip okunabilirlik farkını gör.

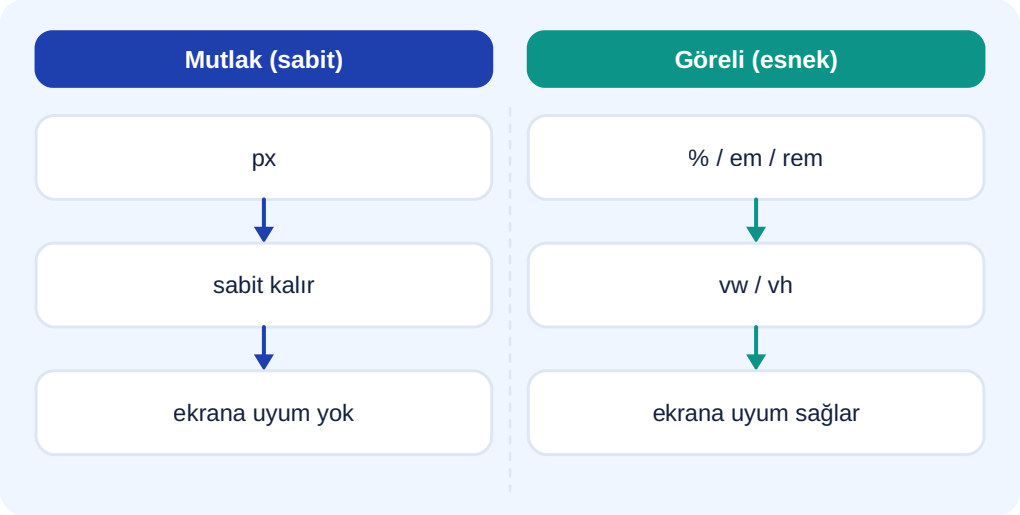
BÖLÜM 06

Birimler

CSS'te boyutları farklı birimlerle ifade edersin. Doğru birimi seçmek, özellikle responsive tasarımda büyük fark yaratır.

Mutlak ve göreli birimler

- **px** — sabit piksel (mutlak).
- **%** — kapsayıcıya göre yüzde.
- **em / rem** — yazı boyutuna göre (rem = kök boyut).
- **vw / vh** — ekran genişliği/yüksekliğine göre.



Şema 6.1 — Mutlak birimler sabittir; göreli birimler ekrana uyum sağlar.

Birim örnekleri

```
.kutu {
  width: 80%;
  padding: 1rem;
  font-size: 1.25rem;
}
```

İPUCU

Responsive tasarımda **göreli birimleri** tercih et: `rem` (boyutlar için), `%` ve `vw` (genişlik için). Her şeyi `px` ile sabitlersen sayfan farklı ekranlara uyum sağlayamaz. `rem`, kullanıcı yazı boyutunu büyüttüğünde de düzgün ölçeklenir (erişilebilirlik).

Stil uygulanınca ne olur? ETKİ

`width: 80%` öğeyi kapsayıcısının %80'i kadar yapar — ekran küçülünce öğe de küçülür. `1rem` ise kök yazı boyutuna (genelde 16px) bağlı, esnek bir ölçüdür.

Alıştırma 8 dk

Birimleri dene:

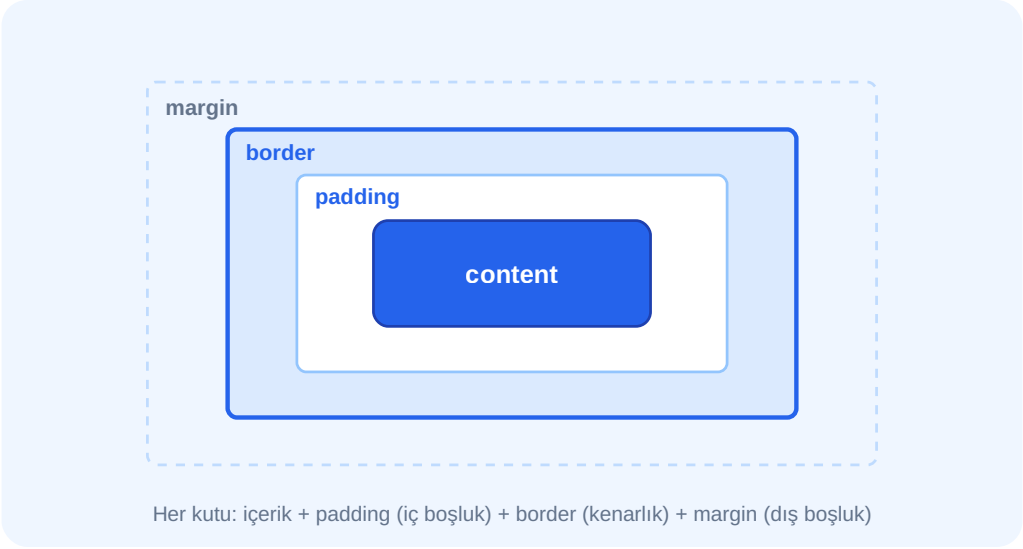
1. Bir öğeye `%` ile genişlik ver, pencereyi daralt.
2. `rem` ile bir font boyutu ayarla.
3. px ve % davranışının farkını gözlemle.

BÖLÜM 07

Kutu Modeli (Box Model)

CSS'te her öğe bir kutudur. Bu kutunun katmanlarını — içerik, iç boşluk, kenarlık, dış boşluk — anlamak, düzen kurmanın temelidir. Bu, CSS'in en önemli kavramlarından biridir.

Kutunun katmanları



Şema 7.1 — Kutu modeli: içerik + padding + border + margin.

- **content:** öğenin asıl içeriği (metin, görsel).
- **padding:** içerik ile kenarlık arasındaki iç boşluk.
- **border:** kutunun kenarlığı.
- **margin:** kutu ile diğer öğeler arasındaki dış boşluk.

Kutu modeli özellikleri

```
.kart {
  padding: 16px;
  border: 2px solid #2563EB;
  margin: 24px;
}
```

İPUCU

`padding` (iç boşluk) ile `margin` (dış boşluk) farkını iyi öğren: padding kutunun **içinde**, içeriği kenarlıktan uzaklaştırır; margin kutunun **dışında**, onu komşularından uzaklaştırır. `box-sizing: border-box` ise boyut hesabını çok kolaylaştırır.

Stil uygulanınca ne olur? ETKİ

Bu kurallarla kart: içeriğin çevresinde 16px iç boşluk, mavi bir kenarlık ve çevresinde 24px dış boşlukla görünür. Boşluklar düzenin nefes almasını sağlar.

Alıştırma 10 dk

Kutuyu şekillendir:

1. Bir öğeye `padding`, `border` ve `margin` ver.
2. padding ve margin değerlerini değiştirip farkı gözlemle.
3. Tarayıcı geliştirici araçlarında kutu modelini incele.

BÖLÜM 08

Görünüm: Kenarlık, Köşe, Gölge

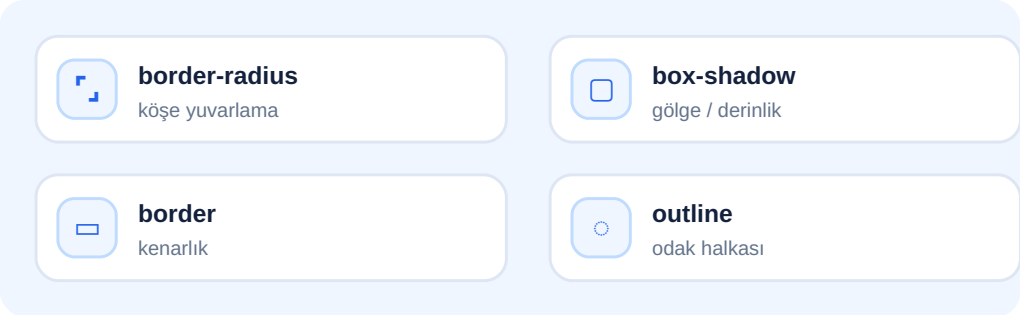
Birkaç basit özellikle öğelerine modern, profesyonel bir görünüm kazandırabilirsin: yuvarlatılmış köşeler, gölgeler ve şık kenarlıklar.

Görünüm özellikleri

- `border-radius` — köşeleri yuvarlatır.
- `box-shadow` — gölge ekler (derinlik hissi).
- `border` — kenarlık stili, kalınlığı, rengi.
- `outline` — kenarlığın dışında, özellikle odak (focus) için.

Modern bir kart görünümü

```
.kart {
  border-radius: 12px;
  box-shadow: 0 4px 12px rgba(0,0,0,0.1);
  background: #fff;
  padding: 20px;
}
```



Şema 8.1 — Görünümü zenginleştiren temel özellikler.

İPUCU

Gölgelerde **abartma**; hafif, yumuşak gölgeler (`rgba` ile düşük opaklık) modern ve zarif görünür. Çok koyu/sert gölgeler eski moda durur. `border-radius` ile köşeleri biraz yuvarlatmak, arayüze sıcaklık katar.

Stil uygulanınca ne olur? ETKİ

Bu kurallarla öğe: yuvarlatılmış köşeli, hafif gölgeli, beyaz bir "kart" olarak görünür — derinlik ve modern bir his verir. Aynı içerik, çok daha profesyonel görünür.

Alıştırma

10 dk

Kart tasarla:

1. Bir `<div>`'i `border-radius` ve `box-shadow` ile karta dönüştür.
2. Gölge değerlerini değiştirip etkisini gör.
3. Hafif bir gölgenin daha şık durduğunu fark et.

SEVİYE 2

Düzen (Layout)

Öğeleri yerleştirmek: display türleri, konumlandırma (position), Flexbox temelleri ve uygulaması, CSS Grid temelleri ve uygulaması.

BÖLÜM 09

Display ve Görünüm Türleri

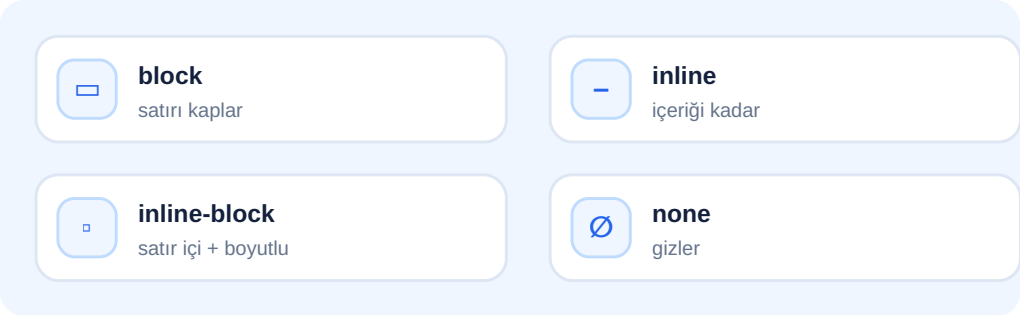
Her öğenin bir "display" türü vardır; bu, öğenin sayfada nasıl yer kapladığını belirler. Düzen kurmanın ilk anahtarı bu kavramı anlamaktır.

Temel display değerleri

- `block` — tüm satırı kaplar (div, p, h1…).
- `inline` — sadece içeriği kadar yer kaplar (span, a…).
- `inline-block` — satır içi durur ama boyut alabilir.
- `none` — öğeyi tamamen gizler.

display değerleri

```
 .menu a   { display: inline-block; }
.gizli    { display: none; }
.kapsayici{ display: flex; }   /* sıradaki konu */
```



Şema 9.1 — Display türleri öğenin nasıl yer kapladığını belirler.

İPUCU

`display: none` öğeyi tamamen kaldırır (yerini de bırakmaz); `visibility: hidden` ise gizler ama yerini korur. Modern düzenlerin çoğu `display: flex` veya `display: grid` üstüne kuruludur — sıradaki bölümlerde bunları öğreneceksin.

Stil uygulanınca ne olur? ETKİ

`block` öğeler alt alta dizilir; `inline` öğeler yan yana akar. `display: none` verilen öğe ekranda hiç görünmez ve yer kaplamaz.

Alıştırma

8 dk

Display'i dene:

1. Bir `<span>`'i `display: block` yapıp farkı gör.
2. Menü bağlantılarını `inline-block` yap.
3. Bir öğeyi `display: none` ile gizle.

BÖLÜM 10

Konumlandırma (Position)

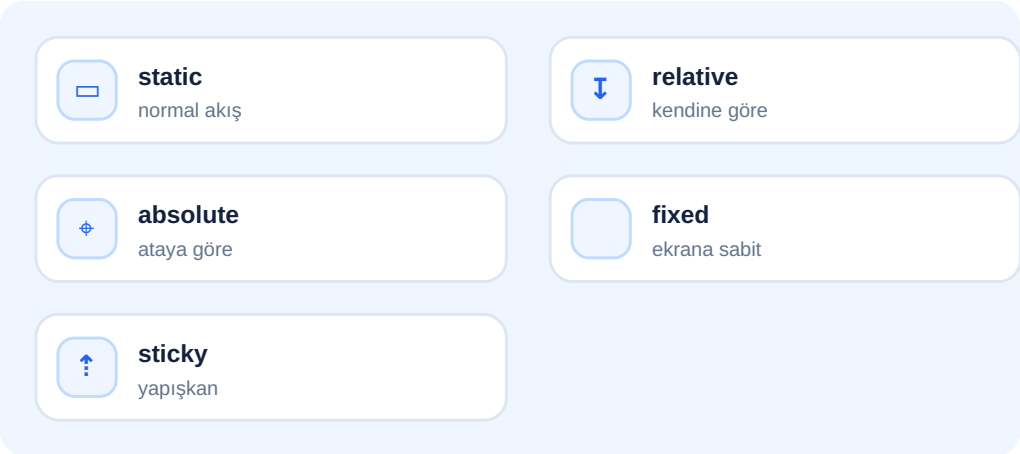
Bazen bir öğeyi normal akışın dışına çıkarıp tam istediğin yere koymak istersin: sabit bir başlık, köşede bir rozet, kaydırınca yapışan bir menü. İşte `position` bunun içindir.

Position değerleri

- `static` — varsayılan, normal akış.
- `relative` — kendi yerine göre kaydırılabilir.
- `absolute` — en yakın konumlu ataya göre yerleşir.
- `fixed` — ekrana sabitlenir (kaydırınca durur).
- `sticky` — belli noktaya gelince yapışır.

Sabit ve yapışkan konum

```
 .ust-bar  { position: fixed; top: 0; }
.menu      { position: sticky; top: 0; }
.rozet     { position: absolute; top: 8px; right: 8px; }
```



Şema 10.1 — Position değerleri öğeyi nasıl konumlandırır.

İPUCU

`position: absolute` bir öğeyi en yakın **konumlu** (position'ı static olmayan) atasına göre yerleştirir; genelde ataya `position: relative` verip içine absolute öğe koyarsın. `sticky` ise menüleri kaydırınca yapıştırmak için harikadır.

Stil uygulanınca ne olur? ETKİ

`fixed` bir bar, sayfayı kaydırsan bile ekranın üstünde sabit kalır. `sticky` bir menü, normal akışta durur ama üst kenara gelince yapışıp orada kalır.

Alıştırma 10 dk

Konumlandır:

1. Bir üst barı `position: fixed` yap.
2. Bir kartın köşesine `absolute` ile rozet koy (ata `relative`).
3. Bir menüyü `sticky` yapıp kaydır.

BÖLÜM 11

Flexbox: Temeller

Flexbox (esnek kutu), öğeleri bir eksen boyunca hizalamanın modern ve kolay yoludur: yan yana dizmek, ortalamak, eşit aralıklarla yaymak. Düzen kurmanın en çok kullanılan araçlarından biridir.

Temel kavramlar

- Kapsayıcıya `display: flex` verirsin; içindekiler "flex öğesi" olur.
- `flex-direction` — ana eksen yönü (row / column).
- `justify-content` — ana eksende hizalama.
- `align-items` — çapraz eksende hizalama.

Şema 11.1 — Flex kapsayıcı: ana eksen (justify-content) ve çapraz eksen (align-items).

Öğeleri ortalama

```
.kapsayici {
  display: flex;
  justify-content: center;  /* yatay ortala */
  align-items: center;      /* dikey ortala */
  gap: 16px;                /* öğeler arası boşluk */
}
```

İPUCU

Bir şeyi tam ortalamak (hem yatay hem dikey) eskiden CSS'in en sinir bozucu işiydi; Flexbox ile artık üç satır: `display:flex; justify-content:center; align-items:center;`. `gap` özelliği de öğeler arası boşluğu zahmetsizce verir.

Stil uygulanınca ne olur? ETKİ

`justify-content: center` öğeleri yatayda ortalar; `align-items: center` dikeyde ortalar. İkisi birlikte, öğeleri kapsayıcının tam ortasına yerleştirir.

Alıştırma 12 dk

Flex dene:

1. Bir kapsayıcıyı `display: flex` yap.
2. İçindeki öğeleri `justify-content` ile yatayda hizala.
3. `gap` ile öğeler arasına boşluk koy.

BÖLÜM 12

Flexbox: Uygulama

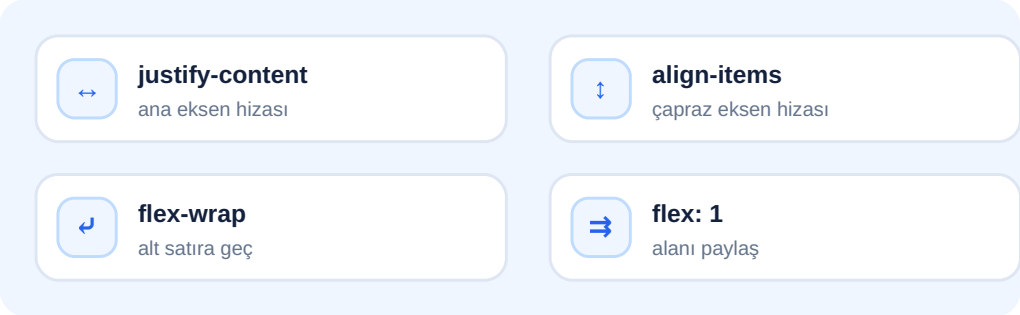
Flexbox'ı gerçek düzenlerde kullanalım: gezinme çubukları, kart sıraları, eşit yükseklikli sütunlar. Birkaç özellikle çok güçlü düzenler kurulur.

Sık kullanılan flex ayarları

- `justify-content: space-between` — öğeleri uçlara yayar (navbar için ideal).
- `flex-wrap: wrap` — sığmazsa alt satıra geçirir (responsive).
- `flex: 1` — öğenin kalan alanı eşit paylaşmasını sağlar.

Gezinme çubuğu (navbar)

```
.navbar {
  display: flex;
  justify-content: space-between;
  align-items: center;
}
.kartlar { display: flex; flex-wrap: wrap; gap: 20px; }
```



Şema 12.1 — Gerçek düzenlerde sık kullanılan flex özellikleri.

İPUCU

`justify-content: space-between` bir navbar'da logoyu sola, menüyü sağa itmenin en temiz yoludur. `flex-wrap: wrap` + `gap` ise kart ızgaralarını responsive yapmanın kolay yoludur — ekran daralınca kartlar alt satıra akar.

Stil uygulanınca ne olur? ETKİ

`space-between` öğeleri iki uca yaslayıp aralarına eşit boşluk koyar (navbar görünümü). `flex-wrap: wrap` sayesinde dar ekranda kartlar otomatik alt satıra geçer.

Alıştırma

12 dk

Düzen kur:

1. `space-between` ile bir navbar yap.
2. `flex-wrap` ile responsive bir kart sırası kur.
3. Pencereyi daraltıp kartların alt satıra geçtiğini gör.

BÖLÜM 13

CSS Grid: Temeller

Grid (ızgara), iki boyutlu (satır + sütun) düzenler için tasarlanmıştır. Flexbox tek eksende güçlüyken, Grid tüm sayfa düzenlerini kurmak için idealdir.

Temel kavramlar

- Kapsayıcıya `display: grid` verirsin.
- `grid-template-columns` — sütun sayısı ve genişlikleri.
- `gap` — hücreler arası boşluk.
- `fr` birimi — kalan alanı oransal paylaştırır.

Şema 13.1 — Grid: satır ve sütunlardan oluşan iki boyutlu düzen.

Üç eşit sütunlu ızgara

```
.izgara {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;  /* 3 eşit sütun */
  gap: 16px;
}
```

İPUCU

`fr` ("fraction") birimi Grid'in sihiridir: `1fr 1fr 1fr` kalan alanı üç eşit parçaya böler. `repeat(3, 1fr)` ile aynısını kısaca yazabilirsin. Grid, "şu kadar sütun olsun" demenin en doğrudan yoludur.

Stil uygulanınca ne olur? ETKİ

`grid-template-columns: 1fr 1fr 1fr` içeriği üç eşit genişlikte sütuna yerleştirir; `gap` aralarına boşluk koyar. Öğeler otomatik olarak hücrelere dizilir.

Alıştırma

12 dk

Grid kur:

1. Bir kapsayıcıyı `display: grid` yap.
2. `grid-template-columns` ile üç sütun tanımla.
3. `gap` ile hücreler arasına boşluk koy.

BÖLÜM 14

Grid: Uygulama

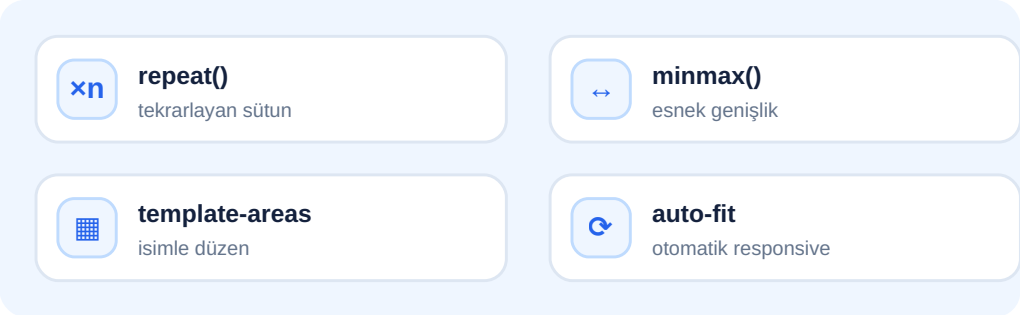
Grid'in asıl gücü, tüm sayfa düzenini (başlık, kenar çubuğu, ana içerik, altbilgi) tek bir ızgarayla kurabilmektir. Karmaşık düzenler birden basitleşir.

Güçlü grid özellikleri

- `repeat()` — tekrar eden sütunları kısaca yaz.
- `minmax()` — esnek ama sınırlı genişlikler.
- `grid-template-areas` — düzeni isimlerle "çiz".
- `auto-fit / auto-fill` — otomatik responsive ızgaralar.

Otomatik responsive kart ızgarası

```
.galeri {
  display: grid;
  grid-template-columns:
    repeat(auto-fit, minmax(220px, 1fr));
  gap: 20px;
}
```



Şema 14.1 — Gelişmiş grid özellikleriyle responsive düzenler.

İPUCU

`repeat(auto-fit, minmax(220px, 1fr))` tek satırla **responsive bir ızgara** kurar: her kart en az 220px olur, ekrana sığdığı kadar sütun açılır, kalanı alt satıra geçer — medya sorgusu bile gerekmez. Bu, modern CSS'in en kullanışlı kalıplarından biridir.

Stil uygulanınca ne olur? ETKİ

Bu kural, ekran genişledikçe daha çok sütun, daraldıkça daha az sütun gösterir; kartlar her zaman en az 220px kalır. Tek satır CSS ile tamamen responsive bir galeri elde edersin.

Alıştırma

12 dk

Sayfa düzeni kur:

1. `repeat(auto-fit, minmax(...))` ile responsive bir kart ızgarası yap.
2. Pencereyi daraltıp sütun sayısının değiştiğini gör.
3. `gap` ile aralıkları ayarla.

SEVİYE 3

Responsive ve Etkileşim

Modern ve canlı arayüzler: responsive tasarım, medya sorguları, geçişler, dönüşümler, animasyonlar ve CSS değişkenleri.

BÖLÜM 15

Responsive Tasarım

Sayfan telefonda, tablette ve masaüstünde iyi görünmeli. Responsive (duyarlı) tasarım, aynı sayfanın her ekrana uyum sağlamasıdır. Bugün bu bir lüks değil, zorunluluktur.

Responsive ilkeleri

- **Esnek birimler** kullan (%, rem, fr) — sabit px yerine.
- **Mobil öncelikli** düşün: önce küçük ekran, sonra büyüt.
- `<meta name="viewport">` olmadan responsive çalışmaz (HTML modülünden).
- Flexbox/Grid + `flex-wrap` / `auto-fit` doğal responsive verir.

Şema 15.1 — Aynı sayfa, ekrana göre farklı düzen: responsive tasarım.

Esnek, responsive temel

```
.kapsayici { width: 90%; max-width: 1100px; margin: 0 auto; }
img { max-width: 100%; height: auto; }
```

İPUCU

`img { max-width: 100%; }` küçük ama altın değerinde bir kuraldır: görsellerin kapsayıcıdan taşmasını önler, böylece dar ekranlarda sayfa bozulmaz. `max-width` + `margin: 0 auto` ise içeriği büyük ekranlarda ortalar ve aşırı genişlemesini engeller.

Stil uygulanınca ne olur? ETKİ

Esnek birimler ve `max-width: 100%` ile sayfa, ekran küçüldükçe içeriği yeniden düzenler; görseller taşmaz, metin okunur kalır. Aynı HTML, her cihazda düzgün görünür.

Alıştırma

10 dk

Responsive yap:

1. Bir kapsayıcıya `width: 90%; max-width` ve `margin: 0 auto` ver.
2. Görsellere `max-width: 100%` ekle.
3. Pencereyi daraltıp düzenin uyum sağladığını gör.

BÖLÜM 16

Medya Sorguları (Media Queries)

Bazen sadece esnek birimler yetmez; belli ekran boyutlarında düzeni tamamen değiştirmek istersin. Medya sorguları (`@media`) tam da bunu yapar: "ekran şu kadar darsa, şu stilleri uygula".

Nasıl çalışır?

- `@media` bir koşul belirtir (örn. en fazla 600px genişlik).
- Koşul sağlanırsa içindeki kurallar devreye girer.
- Bu "kırılım noktalarında" (breakpoint) düzen değişir.

Mobil için tek sütun

```
 .izgara { display: grid; grid-template-columns: 1fr 1fr 1fr; }

@media (max-width: 600px) {
  .izgara { grid-template-columns: 1fr; }  /* mobilde tek sütun */
}
```

Şema 16.1 — Medya sorgusu: ekran boyutuna göre stil değiştirir.

İPUCU

Mobil öncelikli yaz: önce küçük ekran stillerini ver, sonra `@media (min-width: ...)` ile büyük ekranlara ekleme yap. Bu yaklaşım daha temiz ve performanslıdır. Yaygın kırılım noktaları telefon, tablet ve masaüstü çevresindedir ama içeriğine göre seç.

Stil uygulanınca ne olur? ETKİ

`@media (max-width: 600px)` içindeki kurallar yalnızca ekran 600px veya daha darken uygulanır. Böylece masaüstünde üç sütunlu ızgara, telefonda tek sütuna düşer.

Alıştırma 12 dk

Kırılım ekle:

1. Çok sütunlu bir düzen yap.
2. Bir `@media (max-width: 600px)` ile mobilde tek sütuna düşür.
3. Pencereyi daraltıp değişimi gör.

BÖLÜM 17

Geçişler (Transitions)

Ani değişiklikler sert görünür; yumuşak geçişler arayüze zarafet katar. `transition`, bir özelliğin değerinin zamanla, akıcı biçimde değişmesini sağlar — özellikle `:hover` ile.

Geçiş mantığı

- `transition`, hangi özelliğin ne kadar sürede değişeceğini söyler.
- `:hover` gibi durum değişiklikleri yumuşar.
- Renk, boyut, gölge gibi pek çok özellik animasyonlanabilir.

Yumuşak hover efekti

```
.dugme {
  background: #2563EB;
  transition: background 0.2s, transform 0.2s;
}
.dugme:hover {
  background: #1E40AF;
  transform: translateY(-2px);
}
```

Şema 17.1 — transition ile durum değişiklikleri yumuşar.

İPUCU

Geçiş sürelerini **kısa tut** (0.15–0.3s); uzun geçişler arayüzü yavaş ve hantal hissettirir. Her şeyi animasyonlamak yerine (`transition: all`) yalnızca değişen özellikleri belirtmek daha performanslıdır. Küçük dokunuşlar büyük fark yaratır.

Stil uygulanınca ne olur? ETKİ

`transition` sayesinde düğmenin üzerine gelince rengi ve konumu aniden değil, 0.2 saniyede yumuşakça değişir. Bu küçük hareket, arayüzü canlı ve dokunulası hissettirir.

Alıştırma 8 dk

Geçiş ekle:

1. Bir düğmeye `:hover` rengi ver.
2. `transition` ile bu değişimi yumuşat.
3. Süreyi değiştirip his farkını gözlemle.

BÖLÜM 18

Dönüşümler (Transforms)

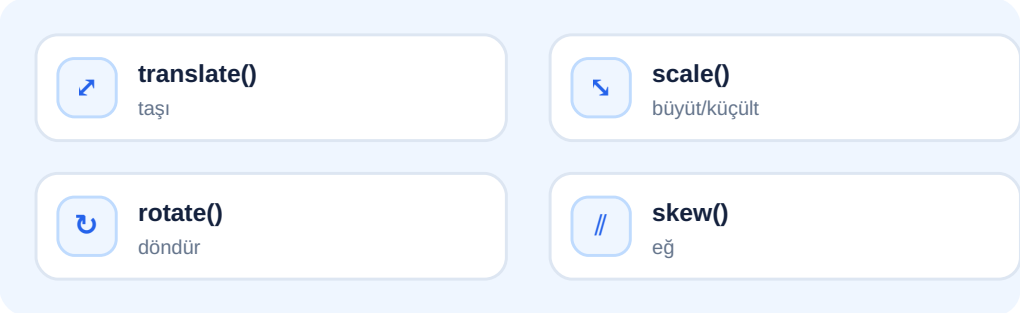
`transform`, öğeleri taşımak, büyütmek, döndürmek ve eğmek için kullanılır — hem statik düzenlemelerde hem de animasyonlarda. Sayfa akışını bozmadan görsel oyunlar yapar.

Dönüşüm türleri

- `translate()` — taşı (x/y).
- `scale()` — büyüt/küçült.
- `rotate()` — döndür.
- `skew()` — eğ.

Üzerine gelince büyüt

```
.kart { transition: transform 0.2s; }
.kart:hover { transform: scale(1.05); }
.rozet { transform: rotate(-8deg); }
```



Şema 18.1 — Transform fonksiyonları.

İPUCU

`transform` öğeyi görsel olarak değiştirir ama **sayfa akışını bozmaz** (yeri sabit kalır), bu yüzden animasyonlar için idealdir ve performanslıdır. `transition` ile birleştirince akıcı hover efektleri (büyüme, hafif dönme) elde edersin.

Stil uygulanınca ne olur? ETKİ

`scale(1.05)` öğeyi %5 büyütür; `:hover` ve `transition` ile birleşince, üzerine gelince yumuşakça büyüyen bir kart elde edersin. `rotate` ise öğeyi belirtilen açıda döndürür.

Alıştırma

8 dk

Dönüştür:

1. Bir karta `:hover` + `transform: scale()` ekle.
2. Bir öğeyi `rotate()` ile hafifçe döndür.
3. `transition` ile dönüşümü yumuşat.

BÖLÜM 19

Animasyonlar

Geçişler bir durumdan diğerine yumuşak geçer; animasyonlar ise kendi başına, çok aşamalı hareketler oluşturur (örn. dönen bir yükleme simgesi, beliren bir uyarı). `@keyframes` ile kareleri tanımlarsın.

İki parça

- `@keyframes` — animasyonun karelerini (0%, 50%, 100%) tanımlar.
- `animation` — bu kareleri bir öğeye uygular (süre, tekrar...).

Beliren bir öğe

```
@keyframes belir {
  from { opacity: 0; transform: translateY(10px); }
  to   { opacity: 1; transform: translateY(0); }
}
.uyari { animation: belir 0.4s ease-out; }
```

Şema 19.1 — @keyframes: animasyonun başı, ortası ve sonu.

İPUCU

Animasyonları **amaçlı** kullan; her şeyi oynatmak dikkat dağıtır ve yorar. Yükleme göstergeleri, hafif beliriş efektleri ve geri bildirimler için idealdir. Erişilebilirlik için `prefers-reduced-motion`'a saygı göster (Seviye 4) — bazı kullanıcılar hareketten rahatsız olur.

Stil uygulanınca ne olur? ETKİ

Bu animasyonla `.uyari` öğesi, görünür olduğunda hafifçe aşağıdan yukarı kayarak ve belirerek gelir (0.4 saniyede). İçerik aynıdır; ama beliriş, dikkati nazikçe çeker.

Alıştırma 10 dk

Animasyon yap:

1. Bir `@keyframes` tanımla (örn. beliren bir öğe).
2. Onu bir öğeye `animation` ile uygula.
3. Süre ve `ease` değerini değiştirip his farkını gör.

BÖLÜM 20

Değişkenler ve Yeniden Kullanım

Aynı rengi veya boşluğu onlarca yerde tekrarlamak yerine, bir kez tanımlayıp her yerde kullanırsın. CSS değişkenleri (custom properties), kodunu düzenli ve kolay değiştirilebilir kılar.

Değişken mantığı

- `:root` içinde `--ad: değer;` ile tanımlarsın.
- `var(--ad)` ile her yerde kullanırsın.
- Tek bir yerden değiştirince, kullanıldığı her yer güncellenir.

Renk ve boşluk değişkenleri

```
:root {
  --ana-renk: #2563EB;
  --bosluk: 16px;
}
.dugme { background: var(--ana-renk); padding: var(--bosluk); }
.baslik { color: var(--ana-renk); }
```

Şema 20.1 — CSS değişkeni: bir kez tanımla, her yerde kullan, tek yerden değiştir.

İPUCU

Değişkenler özellikle **tema** ve **tutarlılık** için harikadır: ana rengini, boşluklarını, köşe yarıçaplarını değişkenlerde topla. Markanın rengini değiştirmek istediğinde tek satırı düzeltmen yeter. Bu, bu eğitim serisindeki sitelerin de kullandığı bir yaklaşımdır.

Stil uygulanınca ne olur? ETKİ

`--ana-renk`'i bir kez tanımlayıp `var()` ile her yerde kullanınca, o değeri değiştirdiğinde düğme, başlık ve diğer her yer aynı anda güncellenir. Tek kaynaktan tutarlılık sağlanır.

Alıştırma 10 dk

Değişken kullan:

1. `:root` içinde bir renk değişkeni tanımla.
2. Onu `var()` ile birkaç yerde kullan.
3. Değişkenin değerini değiştirip her yerin güncellendiğini gör.

SEVİYE 4

Profesyonel CSS

Ustalık: özgüllük ve cascade, düzenli CSS yazmak, modern düzen teknikleri, erişilebilir ve performanslı CSS, çerçevelere bakış ve responsive bir sayfayı stillemek.

BÖLÜM 21

Özgüllük ve Cascade

Aynı öğeye birden çok kural uyduğunda hangisi kazanır? Bunun cevabı "cascade" (kademe) ve "özgüllük" (specificity) kurallarındadır. Bu kavram, "neden stilim uygulanmıyor?" sorusunun en sık cevabıdır.

Kim kazanır?

- Daha **özgül** seçici kazanır: id > class > element.
- Eşitlikte, sonra gelen (daha aşağıdaki) kural kazanır.
- `!important` her şeyi ezer — ama ondan kaçın.

Şema 21.1 — Özgüllük merdiveni: yukarı çıktıkça kural daha güçlü.

Özgüllük çatışması

```
p { color: black; }          /* zayıf */
.uyari { color: orange; }    /* daha güçlü */
#ozel { color: red; }        /* en güçlü */
```

İPUCU

"Stilim neden uygulanmıyor?" sorusunun cevabı genelde özgüllüktür: daha güçlü bir seçici onu eziyordur. Çözüm `!important` eklemek **değildir** (bu bir sonraki sorunu büyütür); bunun yerine seçicilerini sade ve tutarlı tut, çoğunlukla class kullan.

Stil uygulanınca ne olur? ETKİ

Aynı öğeye üç kural da uysa, en özgül olan (`#ozel`) kazanır ve metin kırmızı olur. Özgüllüğü anlamak, beklenmedik stil davranışlarını çözmenin anahtarıdır.

Alıştırma 10 dk

Özgüllüğü test et:

1. Aynı öğeye element, class ve id ile farklı renkler ver.
2. Hangisinin kazandığını gözlemle.
3. Neden `!important` 'tan kaçınmak gerektiğini yaz.

BÖLÜM 22

Düzenli CSS Yazmak

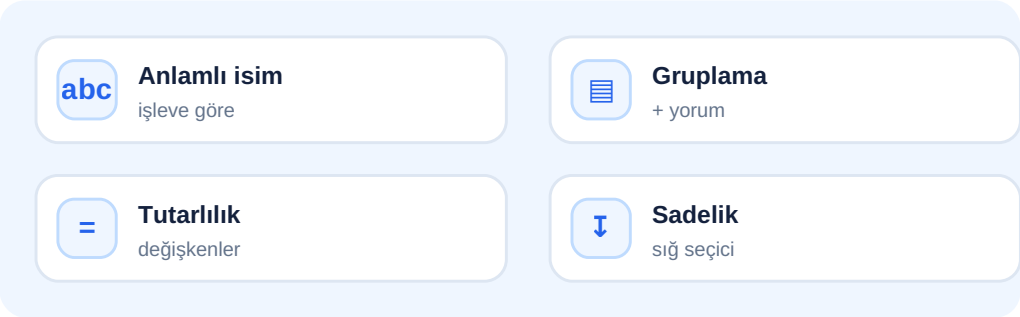
CSS büyüdükçe dağınıklaşmaya meyillidir. Birkaç alışkanlık, kodunu okunur, tutarlı ve bakımı kolay tutar — hem kendin hem ekibin için.

Temiz CSS alışkanlıkları

- **Anlamlı class adları:** `.kart-baslik`, `.btn-birincil` (renk değil işlev).
- **Gruplama:** ilgili kuralları bir arada tut, yorumla böl.
- **Tutarlılık:** aynı boşluk/renk değerlerini (değişkenlerle) kullan.
- **Sadelik:** gereksiz derin/özgül seçicilerden kaçın.

Anlamlı ve düzenli

```
/* --- Kartlar --- */
.kart { padding: var(--bosluk); border-radius: 12px; }
.kart-baslik { font-weight: 700; }
.kart-metin  { color: #475569; }
```



Şema 22.1 — Sürdürülebilir CSS'in temel alışkanlıkları.

İPUCU

Class adlarını **işleve göre** ver, görünüme göre değil: `.uyari` (iyi) yerine `.kirmizi-yazi` (kötü). Çünkü yarın uyarıyı turuncu yaparsan, `.kirmizi-yazi` adı yalan olur. İyi isimlendirme, kodun yıllar sonra bile anlaşılır kalmasını sağlar.

Stil uygulanınca ne olur? ETKİ

Düzenli CSS görünümü değiştirmez; ama kodu okumayı, bir şeyi bulup değiştirmeyi ve ekipçe çalışmayı kolaylaştırır. Dağınık CSS çalışır ama zamanla bakılamaz hâle gelir.

Alıştırma

8 dk

Düzene sok:

1. Dağınık birkaç kuralı yorumlarla grupla.
2. Görünüme dayalı bir class adını işleve göre yeniden adlandır.
3. Tekrar eden değerleri bir değişkene taşı.

BÖLÜM 23

Modern Düzen Teknikleri

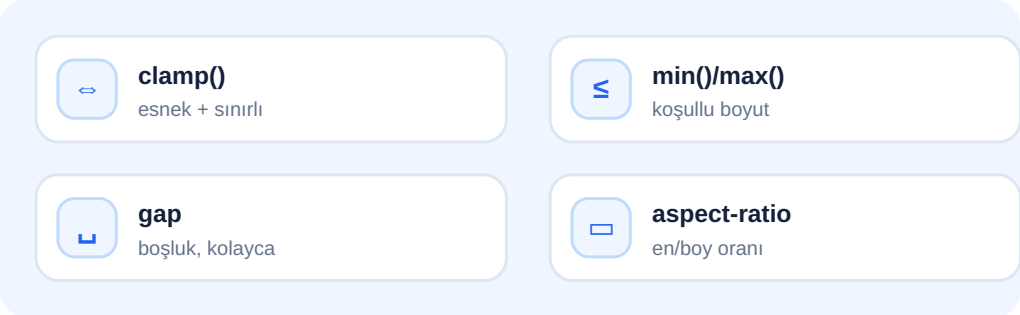
Son yıllarda CSS'e güçlü, hayat kolaylaştıran işlevler eklendi. Bunlar, eskiden JavaScript veya karmaşık hesap gerektiren işleri tek satıra indirir.

İşine yarayacak modern özellikler

- `clamp(min, tercih, max)` — esnek ama sınırlı boyut (responsive tipografi).
- `min()` / `max()` — koşullu boyutlar.
- `gap` — flex ve grid'de boşluk (margin derdine son).
- `aspect-ratio` — en-boy oranını sabitle (örn. 16/9).

Akıcı (fluid) başlık boyutu

```
h1 { font-size: clamp(1.8rem, 5vw, 3rem); }
.video { aspect-ratio: 16 / 9; }
```



Şema 23.1 — Modern CSS işlevleri karmaşık işleri basitleştirir.

İPUCU

`clamp(1.8rem, 5vw, 3rem)` başlık boyutunu ekrana göre akıcı biçimde ayarlar: çok küçülmez (en az 1.8rem), çok büyümez (en fazla 3rem), arada ekranla ölçeklenir. Bu tek satır, eskiden birkaç medya sorgusu gerektiren işi halleder.

Stil uygulanınca ne olur? ETKİ

`clamp()` ile başlık, ekran genişledikçe yumuşakça büyür ama belirlediğin sınırların dışına çıkmaz. `aspect-ratio` ise bir kutuyu (örn. video) her zaman 16:9 oranında tutar.

Alıştırma

8 dk

Modern teknik dene:

1. Bir başlığa `clamp()` ile akıcı boyut ver.
2. Bir kutuya `aspect-ratio` uygula.
3. Pencereyi yeniden boyutlandırıp etkilerini gör.

BÖLÜM 24

Erişilebilir ve Performanslı CSS

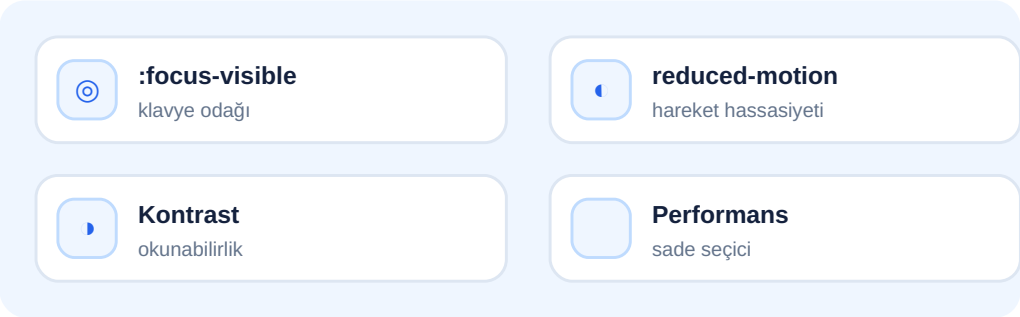
İyi CSS yalnızca güzel değil; herkes için kullanılabilir ve hızlıdır. Birkaç ilke, sayfanı daha erişilebilir ve performanslı kılar.

Erişilebilirlik ve performans

- **Görünür odak:** `:focus-visible` ile klavye kullanıcıları için odak halkasını koru.
- **Hareket hassasiyeti:** `prefers-reduced-motion`'a saygı göster.
- **Kontrast:** metin/arka plan kontrastı yeterli olsun.
- **Performans:** aşırı derin seçici ve gereksiz animasyondan kaçın.

Hareketi azaltmaya saygı

```
@media (prefers-reduced-motion: reduce) {
  * { animation: none; transition: none; }
}
a:focus-visible { outline: 2px solid #2563EB; }
```



Şema 24.1 — Erişilebilir ve performanslı CSS ilkeleri.

İPUCU

Odak halkasını (`outline`) **asla sebepsiz kaldırma**; klavyeyle gezen kullanıcılar nerede olduklarını ondan anlar. Çirkin buluyorsan kaldırmak yerine güzelce stille. `prefers-reduced-motion` ise vestibüler rahatsızlığı olan kullanıcılar için animasyonları kapatmanı sağlar — küçük bir özen, büyük bir saygı.

Stil uygulanınca ne olur? ETKİ

Bu kurallarla, hareket azaltmayı tercih eden kullanıcılarda animasyonlar kapanır; klavyeyle gezenler ise hangi öğede olduklarını net bir odak halkasından görür. Sayfa herkes için daha kullanılabilir olur.

Alıştırma

8 dk

Erişilebilir yap:

1. Bağlantılara güzel bir `:focus-visible` stili ver.
2. `prefers-reduced-motion` ile animasyonları azalt.
3. Bir metin/arka plan kontrastını kontrol et.

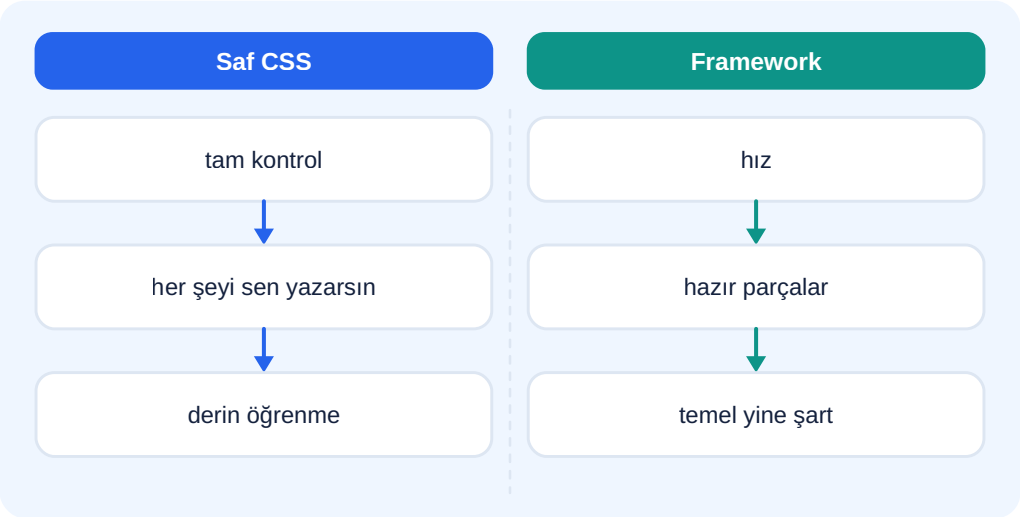
BÖLÜM 25

CSS Çerçevelerine Bakış

Saf CSS'i öğrendikten sonra, işleri hızlandıran CSS çerçeveleriyle (framework) tanışırsın. Ama önce temeli bilmek şarttır; çerçeveler temelin yerine değil, üstüne gelir.

Yaygın yaklaşımlar

- **Utility-first (Tailwind):** küçük yardımcı sınıflarla hızlı stilleme.
- **Bileşen kütüphaneleri (Bootstrap):** hazır düğme, kart, ızgara.
- Her ikisi de zaman kazandırır ama saf CSS bilgisi gerektirir.



Şema 25.1 — Saf CSS tam kontrol verir; çerçeveler hız katar.

İPUCU

Çerçeveye geçmeden önce **saf CSS'i öğren**; aksi hâlde çerçeve sihir gibi görünür ve bir şey bozulunca düzeltemezsin. Temeli bilen biri için çerçeveler güçlü bir hızlandırıcıdır; bilmeyen için ise bir kara kutudur. Önce neyi otomatikleştirdiğini anla.

Stil uygulanınca ne olur? ETKİ

Çerçeveler görünümü hızlandırır (hazır sınıflar/bileşenler), ama altında hep aynı CSS çalışır. Bu yüzden bu modülde öğrendiklerin, hangi çerçeveyi kullanırsan kullan, işine yarar.

Alıştırma

8 dk

Çerçeveleri tanı:

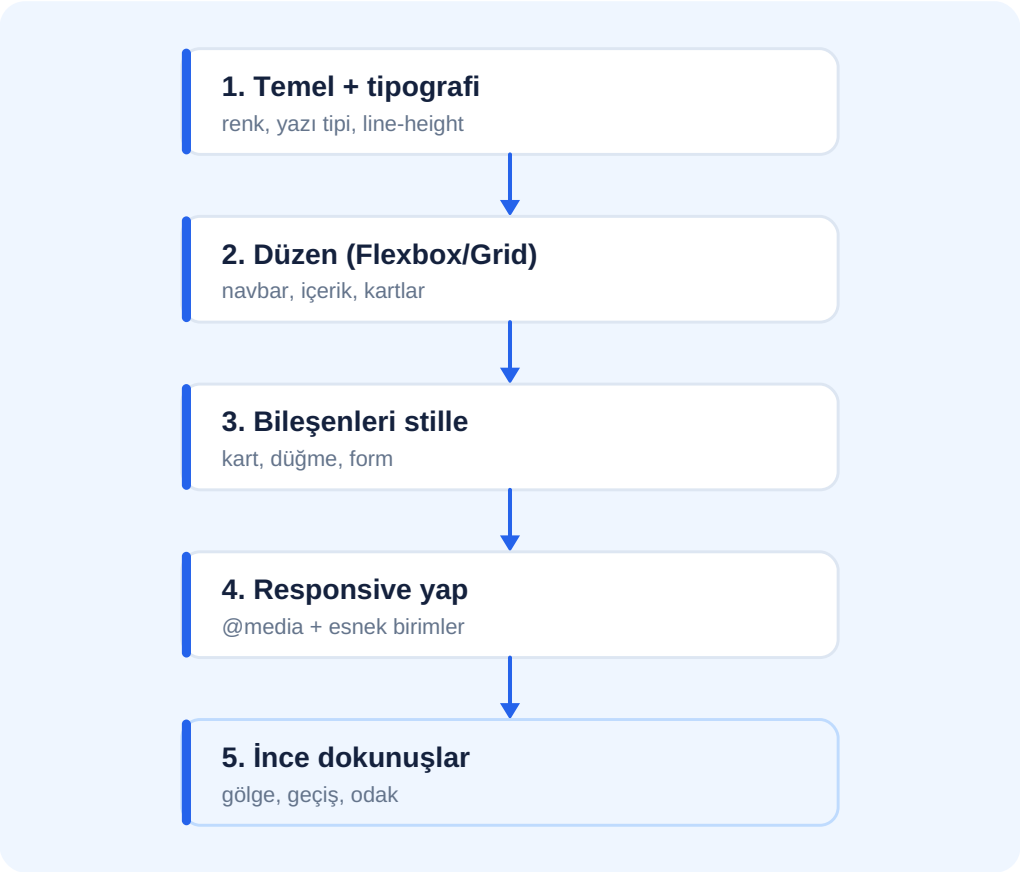
1. Utility-first ve bileşen tabanlı yaklaşımı kendi cümlelerinle açıkla.
2. Bir çerçevenin avantaj ve dezavantajını yaz.
3. Neden önce saf CSS öğrenmek gerektiğini belirt.

BÖLÜM 26

Bitirme: Responsive Bir Sayfayı Stillemek

Tüm CSS bilgini birleştirip, HTML modülünde kurduğun türden bir sayfayı baştan sona stilliyorsun: tipografi, kutu modeli, Flexbox/Grid düzeni ve responsive davranış.

Stilleme akışı



Şema 26.1 — Bir sayfayı sıfırdan stilleme akışı.

Tipik bir başlangıç bloğu

```
 :root { --ana: #2563EB; --bosluk: 16px; }
body { font-family: system-ui; line-height: 1.6; color: #1e293b; }
.kapsayici { width: 90%; max-width: 1100px; margin: 0 auto; }
.navbar { display: flex; justify-content: space-between; align-items: center; }
```

İPUCU

Bir sayfayı stillemeye **genelden özele** doğru başla: önce gövde (tipografi, renk), sonra büyük düzen (Flexbox/Grid), sonra tek tek bileşenler, en son ince dokunuşlar ve responsive. Bu sıra seni dağılmaktan kurtarır. Sıradaki modülde (JavaScript) bu güzel arayüze etkileşim ekleyeceksin.

Stil uygulanınca ne olur?

ETKİ

Bu akışı izleyince HTML modülünde kurduğun sade iskelet, adım adım modern ve responsive bir sayfaya dönüşür: okunaklı tipografi, düzenli yerleşim, şık bileşenler ve her ekrana uyum.

Alıştırma

15 dk

Bir sayfayı stille:

1. Bir HTML sayfasına temel tipografi ve renk ver.
2. Flexbox/Grid ile düzeni kur (navbar + kart ızgarası).
3. Kart ve düğmeleri stille (kutu modeli, gölge, geçiş).
4. `@media` ile mobilde tek sütuna düşür ve test et.

EK

CSS Özellik Sözlüğü

En sık kullanılan CSS özellikleri ve işlevleri. Bir başvuru kaynağı olarak saklayabilirsin.

color / background	`Metin ve arka plan rengi`	font-family / size	`Yazı tipi ve boyut`
line-height	`Satır aralığı`	margin / padding	`Dış / iç boşluk`
border / border-radius	`Kenarlık / köşe`	box-shadow	`Gölge`
display	`block, flex, grid, none`	position	`relative, absolute, fixed`
flex / justify / align	`Flexbox düzeni`	grid-template	`Grid düzeni`
@media	`Responsive kırılım`	var(--x) / :root	`CSS değişkenleri`

CSS'in özeti ETKİ

CSS, **seçicilerle** öğeleri hedefler ve **özellik: değer** kurallarıyla onlara görünüm verir: renk, yazı, boşluk (kutu modeli), düzen (Flexbox/Grid) ve responsive davranış. HTML iskeletine bağlanan iyi bir CSS, aynı içeriği sade bir belgeden modern, kullanışlı bir arayüze dönüştürür.