

WEB PROGRAMLAMA I ■ HAFTA 9

CSS Grid ve Duyarlı Tasarım

■ birimi ■ Media query ■
Mobile-first ■ Kırılma noktaları

Prof. Dr. Özal YILDIRIM

Fırat Üniversitesi — Teknoloji Fakültesi

Yazılım Mühendisliği Bölümü

Flexbox Yeterli Değil mi?

Flexbox, elementleri **tek bir satırda veya sütunda** hizalamak için harikadır. Peki ya **hem satırları hem sütunları aynı anda** kontrol etmek istersek?

↔ Flexbox — Tek Boyut

- ▶ Bir menüdeki linkler
- ▶ Kart içi dikey akış
- ▶ İkon + metin hizalama
- ▶ Buton grubu

İçerik, düzeni belirler.

■ Grid — İki Boyut

- ▶ Sayfanın ana yerleşimi
- ▶ Fotoğraf galerisi
- ▶ Dergi/gazete düzeni
- ▶ Kontrol paneli (dashboard)

Düzen, içeriği belirler.

İkisi rakip değildir — birlikte kullanılırlar. Grid ile sayfa iskeletini kurar, içindeki bileşenleri Flexbox ile dizersiniz.

- ▶ Grid kapsayıcı ve grid ögesi kavramlarını ayırt edebilecek,
- ▶ `grid-template-columns` ve `fr` birimiyle sütun yapısı kurabilecek,
- ▶ `repeat()`, `minmax()` ve `auto-fit` ile otomatik ızgara oluşturabilecek,
- ▶ `grid-template-areas` ile okunabilir sayfa iskeleti tanımlayabilecek,
- ▶ Öğeleri satır/sütun çizgilerine yerleştirebilecek,
- ▶ Duyarlı tasarımın üç temel unsurunu (viewport, akışkan düzen, media query) uygulayabilecek,
- ▶ Mobile-first yaklaşımıyla kırılma noktaları tanımlayabilecek,
- ▶ Yapay zekaya ızgara ürettirip doğrulayabileceksiniz.

Bu Haftanın İçeriği

- 1 CSS Grid Temelleri
- 2 Duyarlı Tasarım
- 3 Yapay Zeka ile Izgara ve Duyarlılık
- 4 Özet ve Değerlendirme



CSS Grid Temelleri

BÖLÜM 1

Grid Kapsayıcısı ve Öğeleri

Tanım

CSS Grid, bir web sayfasını **satır ve sütunlardan oluşan bir ızgaraya** bölmemizi sağlar. Bu ızgara üzerindeki hücelere veya alanlara elementlerimizi **hassas bir şekilde** yerleştirebiliriz.

```
1 .izgara {  
2   display: grid;  
3   grid-template-columns: 200px 1fr 1fr;    /* 3 sutun */  
4   grid-template-rows: auto 1fr auto;       /* 3 satir */  
5   gap: 20px;                               /* satir + sutun ğbosluu */  
6 }
```

Özellik

Görevi

display: grid

Elementi ızgara kapsayıcısına dönüştürür

grid-template-columns

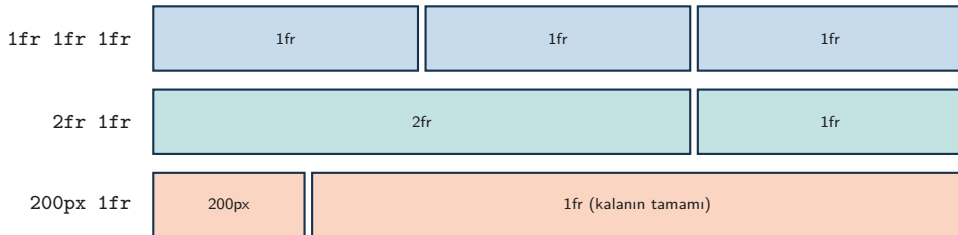
Sütun yapısını tanımlar

Satır yapısını tanımlar

grid-template-rows

fr Birimi: Kesirli Birim

`fr` (*fractional unit*), kapsayıcıdaki **kalan boşluğun payını** temsil eder. Esnek düzenler için son derece kullanışlıdır.



```
grid-template-columns: 2fr 1fr;          /* ana icerik 2 birim, sidebar 1 birim */
grid-template-columns: 200px 1fr;        /* sabit sidebar + esneyen icerik */
grid-template-columns: repeat(3, 1fr);   /* 1fr 1fr 1fr ile ayni */
```

repeat(), minmax() ve Otomatik Izgara

```
1  /* Tekrarlama */
2  grid-template-columns: repeat(4, 1fr);
3  grid-template-columns: repeat(2, 200px) 1fr;    /* karisik */
4
5  /* Alt ve ust sinir */
6  grid-template-columns: repeat(3, minmax(200px, 1fr));
7
8  /* SIHIRLI FORMUL: media query'siz duyarli izgara */
9  .kart-izgara {
10     display: grid;
11     grid-template-columns: repeat(auto-fit, minmax(260px, 1fr));
12     gap: 1.5rem;
13 }
```

Bu Formül Ne Yapar?

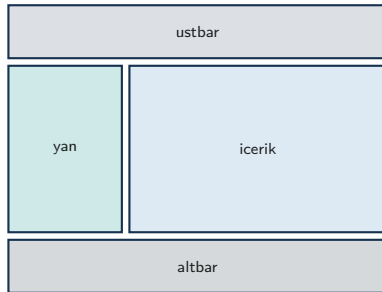
“Her sütun en az 260px olsun; kalan boşluğu paylaşsın. Kaç sütun sığarsa o kadar sütun oluştur.”
1400px’te 5 sütun, 800px’te 3, telefonda 1 — **tek satırda.**

auto-fit vs auto-fill

auto-fit: Boş sütunları **çökertir**, mevcut öğeler alanı doldurur. **auto-fill:** Boş sütunları **korur**, öğeler genişlemez. Genelde **auto-fit** istediğiniz şeydir.

grid-template-areas: Görsel Yerleşim

```
1 .sayfa {
2   display: grid;
3   grid-template-columns: 240px 1fr;
4   grid-template-rows: auto 1fr auto;
5   grid-template-areas:
6     "ustbar  ustbar"
7     "yan     icerik"
8     "altbar  altbar";
9   gap: 1.5rem;
10  min-height: 100vh;
11 }
12
13 header { grid-area: ustbar; }
14 aside  { grid-area: yan;   }
15 main   { grid-area: icerik; }
16 footer { grid-area: altbar; }
```



💡 İpucu

CSS'in en okunaklı özelliği. Kodun kendisi düzenin **resmini çizer**. Boş hücre için nokta kullanılır: "yan ."

Çizgi Numaralarıyla Yerleştirme

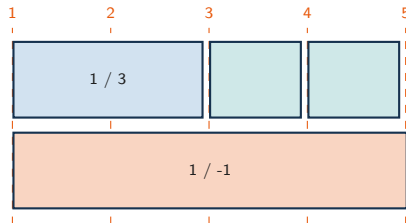
Bir ızgarada **n sütun** varsa **n+1 çizgi** vardır. Öğeleri bu çizgilere göre yerleştirebiliriz.

```
.izgara { display: grid;
          grid-template-columns: repeat(4, 1fr); }

.genis {
  grid-column: 1 / 3;      /* 1. çizgiden 3.'ye */
  grid-column: span 2;     /* 2 sütun kadar */
  grid-column: 1 / -1;     /* baştan SONA kadar */
}

.uzun {
  grid-row: 2 / 4;         /* 2 satır kapsasın */
  grid-row: span 2;
}

/* Kisa yazim: satir-basi / sutun-basi / satir-son / sutun-son */
.ozel { grid-area: 1 / 2 / 3 / 4; }
```



İpucu

1 / -1 en sık kullanılan kalıptır: “**baştan sona tüm sütunları kapla**”.

Izgarada Hizalama

Özellik	Uygulandığı yer	Etkisi
<code>justify-items</code>	Kapsayıcı	Tüm öğeleri hücre içinde yatayda hizalar
<code>align-items</code>	Kapsayıcı	Hücre içinde dikeyde hizalar
<code>place-items</code>	Kapsayıcı	İkisinin kısa yazımı
<code>justify-content</code>	Kapsayıcı	Izgaranın tamamını yatayda hizalar
<code>align-content</code>	Kapsayıcı	Izgarayı dikeyde hizalar
<code>justify-self</code> <code>align-self</code>	Öğe	Tek bir öğeyi hizalar

```
/* Bir şeyi TAM ORTALAMANIN en kısa yolu */
```

```
.orta {  
  display: grid;  
  place-items: center;  
  min-height: 100vh;  
}
```

Uygulama: BookVerse Ana Düzeni

1. Adım — HTML

```
<main>
  <section class="welcome-section">
    <h1>BookVerse'e Hos Geldiniz</h1>
    <p>Burasi, kitaplar hakkında ...</p>
  </section>

  <aside class="sidebar">
    <section>
      <h2>Haftanın Cok Satanlari</h2>
      <table> ... </table>
    </section>
    <section>
      <h2>Bultenimize Kayit Olun</h2>
      <form> ... </form>
    </section>
  </aside>
</main>
```

2. Adım — CSS

```
main {
  display: grid;
  grid-template-columns: 2fr 1fr;
  gap: 30px;
}

.sidebar {
  background-color: #f8f9fa;
  padding: 20px;
  border-radius: 5px;
}
```

İpucu

Ana içerik **2 birim**, kenar çubuğu **1 birim** yer kaplıyor. Yani içerik alanı sidebar'ın **iki katı** genişlikte.

Semantik kazanım: Kenar içeriğini `<aside>` içine aldık — Hafta 4'te öğrendiğimiz gibi bu, ana içerikle **dolaylı ilişkili** içerik içindir.



[Ana Sayfa](#) [Kitaplar](#) [Hakkımızda](#) [İletişim](#)

BookVerse'e Hoş Geldiniz

Burası, kitaplar hakkında yorumlarınızı paylaşılabileceğiniz, yeni yazarlar keşfedebileceğiniz ve kitap tutkunlarıyla buluşabileceğiniz bir platformdur.

**Haftanın Çok
Satanları**

**Bultenimize Kayıt
Olun**

© 2025 BookVerse. Tüm hakları saklıdır.

İma Edilen Izgara (Implicit Grid)

Tanımladığınızdan **daha fazla öge** varsa, tarayıcı otomatik olarak yeni satırlar oluşturur. Bu satırların davranışını kontrol edebilirsiniz.

```
1 .izgara {
2   display: grid;
3   grid-template-columns: repeat(3, 1fr);    /* ACIK (explicit) sutunlar */
4   grid-auto-rows: minmax(120px, auto);      /* IMA EDILEN satirlar */
5   gap: 1rem;
6 }
7
8 /* Ogeleri satir yerine SUTUN yonunde akit */
9 .yatay-akis {
10  display: grid;
11  grid-auto-flow: column;
12  grid-auto-columns: minmax(200px, 1fr);
13 }
14
15 /* Bosluklari otomatik doldur (masonry benzeri) */
16 .sikisik { grid-auto-flow: dense; }
```

grid-auto-rows: minmax(120px, auto) çok kullanışlıdır: “en az 120px olsun, içerik uzunsa

büyüsün”.

Grid ile Geçişler ve İç İç Kullanım

```
1  /* DIS: Grid ile sayfa iskeleti */
2  .sayfa {
3      display: grid;
4      grid-template-columns: 260px 1fr;
5      gap: 2rem;
6  }
7
8  /* IC: Flexbox ile bileşen düzeni */
9  .kitap-kart {
10     display: flex;
11     flex-direction: column;
12     gap: 0.5rem;
13 }
14 .kitap-kart .buton { margin-top: auto; }
15
16 /* Kart ızgarası yine Grid */
17 .kart-alani {
18     display: grid;
19     grid-template-columns:
20         repeat(auto-fit, minmax(240px, 1fr));
21     gap: 1.5rem;
22 }
```



Pratik Kural

Grid → sayfanın iskeleti ve ızgaralar **Flexbox** → bileşenlerin içi ve tek eksenli hizalamalar İkisi iç içe kullanılır. Bir eleman hem grid ögesi hem flex kapsayıcısı olabilir.



Alt Izgara (Subgrid)

`grid-template-columns: subgrid` ile iç ızgara, dış ızgaranın çizgilerini devralır. Kart içi öğelerin kartlar arasında hizalanmasını sağlar. Modern tarayıcılarda desteklenir.



Duyarlı Tasarım

BÖLÜM 2

Duyarlı Tasarım Nedir?

Tanım

Duyarlı Tasarım (Responsive Design), bir web sitesinin yerleşiminin, görüntülendiği cihazın **ekran boyutuna duyarlı** olmasını ve kendini otomatik olarak adapte etmesini sağlayan yaklaşımdır.

1. Viewport

```
<meta  
name="viewport">
```

etiketi **zorunludur**. Olmadan medya sorguları **çalışmaz**.

2. Akışkan Düzen

Sabit px yerine **%**, **fr**, **max-width** ve **minmax()** kullanmak.

3. Media Query

Belirli **kırılma noktalarında** farklı CSS kuralları uygulamak.

Amaç: Mobil kullanıcıların içeriği görmek için sürekli yakınlaştırmasına veya **yatayda kaydırmasına** gerek kalmaması.

Viewport Meta Etiket

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Bu etiket tarayıcıya, sayfanın genişliğini **cihazın ekran genişliğine göre** ayarlamasını söyler.

✗ Etiket Olmadan

Mobil tarayıcılar sayfayı **980px genişliğinde** varsayar ve küçültüp gösterir. Sonuç: Okunamaz derecede küçük metin, sürekli zoom, **medya sorguları hiç tetiklenmez**.

⚠ Bunları YAZMAYIN

```
maximum-scale=1.0  
user-scalable=no
```

Bunlar kullanıcının **yakınlaştırma hakkını** elinden alır — az gören kullanıcılar için ciddi bir **erişilebilirlik ihlalidir** (WCAG 1.4.4).

Media Query Sözdizimi

```
1 @media (min-width: 768px) {
2   /* Ekran 768px VE DAHA GENIS ise uygulanir */
3   .container { width: 960px; margin: 20px auto; }
4 }
5
6 @media (max-width: 767px) {
7   /* Ekran 767px VE DAHA DAR ise uygulanir */
8   .yan-menu { display: none; }
9 }
10
11 /* Aralik */
12 @media (min-width: 768px) and (max-width: 1023px) { ... }
13
14 /* Yon */
15 @media (orientation: landscape) { ... }
16
17 /* Kullanici tercihleri */
18 @media (prefers-color-scheme: dark) { ... }
19 @media (prefers-reduced-motion: reduce) { ... }
20
21 /* Yazdirma */
22 @media print { nav, footer { display: none; } }
```

Önce en küçük ekran için temel stilleri yazın, sonra `min-width` ile büyük ekranlara ekleyin.

✓ Mobile-first

```
/* TEMEL: mobil */
.container { width: auto; margin: 20px 15px; }
main      { display: block; }

/* Geniş ekranlar için EKLE */
@media (min-width: 768px) {
  .container { width: 960px; margin: 20px auto; }
  main { display: grid;
          grid-template-columns: 2fr 1fr;
          gap: 30px; }
}
```

✗ Desktop-first

```
/* TEMEL: masaüstü */
.container { width: 960px; margin: 20px auto; }
main { display: grid;
        grid-template-columns: 2fr 1fr; }

/* Mobilde GERİ AL */
@media (max-width: 767px) {
  .container { width: auto; margin: 20px 15px; }
  main { display: block; }
}
```

💡 İpucu

Neden mobile-first?

- ▶ Mobil cihaz **daha az CSS** indirir ve işler
- ▶ “Geri alma” kodu yazmazsınız — sadece **ekleme**
- ▶ Google **mobile-first indexing** kullanır

Kırılma Noktaları Seçimi

```
/* Yaygın baslangic seti */
@media (min-width: 480px) { } /* buyuk
    telefon */
@media (min-width: 768px) { } /* tablet
    */
@media (min-width: 1024px) { } /* laptop
    */
@media (min-width: 1280px) { } /* genis
    ekran */
```

İpucu

Daha iyi yaklaşım: Cihaz boyutlarını ezberlemeyin. Pencereyi **yavaşça daraltın**; tasarımın *bozulduğu* yere kırılma noktası koyun. *Cihazlar değişir, içeriğiniz değişmez.*

Modern kural: Önce **içsel olarak duyarlı** (intrinsic) çözümleri deneyin; media query'yi **gerektiğinde** ekleyin.

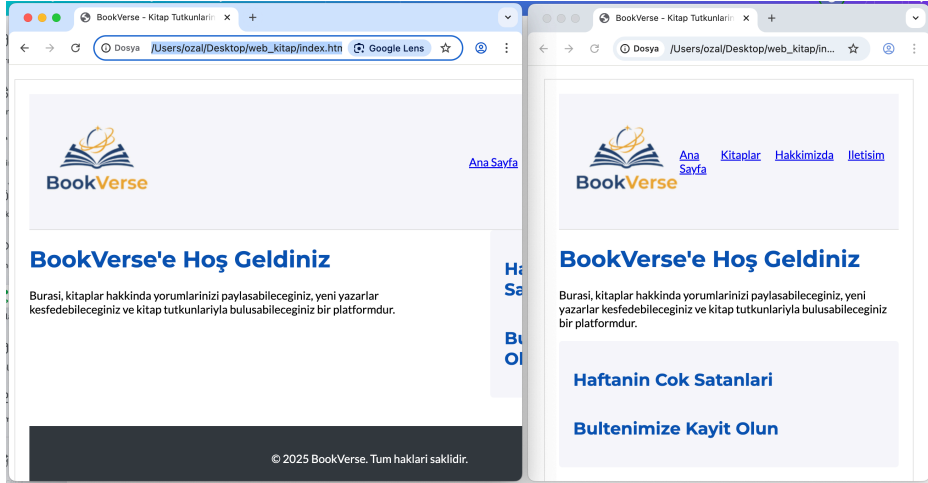


Bazen Hiç Gerekmez

```
/* Media query'siz duyarli ızgara */
.kartlar {
    display: grid;
    grid-template-columns:
        repeat(auto-fit, minmax(260px, 1fr));
    gap: 1.5rem;
}

/* Media query'siz duyarli tipografi */
h1 { font-size: clamp(1.75rem, 5vw, 3rem); }

/* Media query'siz duyarli kapsayici */
.container {
    width: min(1100px, 100% - 2rem);
    margin-inline: auto;
}
```



Duyarlı Görseller

```
img { max-width: 100%; height: auto; display: block; }
```

Bu üç özellik, görsellerin **asla taşmamasını** garanti eder.

```
1 <!-- Farkli ekranlara farkli dosya -->
2 
7
8 <!-- Sanat yonetimi: farkli ekranda farkli KADRAJ -->
9 <picture>
10   <source media="(max-width: 640px)" srcset="hero-mobil.webp">
11   <source media="(min-width: 641px)" srcset="hero-genis.webp">
12   
13 </picture>
```

width ve height yazmayı unutmayın: tarayıcı yeri önceden ayırır ve sayfa yüklenirken **zıplamaz** (layout shift).

Cihaz Modu

Cmd/Ctrl + Shift + M ile açılır.

- ▶ Hazır cihaz profilleri (iPhone, iPad, Galaxy)
- ▶ **Responsive** modda serbest boyutlandırma
- ▶ Üst şeritte **kırılma noktalarınız** görünür
- ▶ Ağ kısıtlama (Slow 3G) ile gerçekçi test
- ▶ Ekran görüntüsü alma

Grid Denetçisi

display: grid olan elementin yanındaki **grid** rozetine tıklayın:

- ▶ Çizgi numaraları görünür
- ▶ Alan adları (**grid-area**) etiketlenir
- ▶ Boşluklar (**gap**) taranır
- ▶ Layout panelinden birden çok ızgara aynı anda izlenebilir

İpucu

Gerçek cihaz testi: Telefonunuzu bilgisayarınızla aynı Wi-Fi'ye bağlayıp, Live Server'ın gösterdiği **ağ adresini** (<http://192.168.x.x:5500>) telefonda açın. Simülasyon her zaman gerçeği yansıtmaz.

Duyarlı Menü: Dikeyden Yataya

```
1  /* --- TEMEL (MOBİL): dikey menu --- */
2  .ust-bar      { display: flex; flex-direction: column; gap: 0.75rem; }
3  .ust-bar nav ul { display: flex; flex-direction: column; gap: 0.5rem;
4                  list-style: none; margin: 0; padding: 0; }
5
6  .ust-bar nav a {
7      display: block;                /* tum satir tiklanabilir olsun */
8      padding: 0.75rem 1rem;        /* dokunma hedefi >= 44px      */
9  }
10
11 /* --- TABLET VE UZERI: yatay menu --- */
12 @media (min-width: 768px) {
13     .ust-bar      { flex-direction: row; align-items: center;
14                     justify-content: space-between; }
15     .ust-bar nav ul { flex-direction: row; gap: 1.5rem; }
16     .ust-bar nav a { padding: 0.5rem 0; }
17 }
```

Dokunma hedefi kuralı: Mobilde tıklanabilir alanlar en az **44×44 piksel** olmalıdır (WCAG 2.5.5 / Apple HIG). `padding` ile büyütün — `margin` ile değil.

Kapsayıcı Sorguları (Container Queries)

Media query **ekranı** ölçer. Ya bileşen, *içinde bulunduğu alana* göre değişmesi gerekiyorsa?

```
1  /* 1) Kapsayiciyi tanimla */
2  .kart-alani {
3      container-type: inline-size;
4      container-name: kartlar;
5  }
6
7  /* 2) Kapsayicinin genisligine gore stil ver */
8  @container kartlar (min-width: 420px) {
9      .kitap-kart {
10         display: grid;
11         grid-template-columns: 120px 1fr;    /* genis alanda yatay kart */
12         gap: 1rem;
13     }
14 }
```

İpucu

Neden devrim? Aynı kart bileşeni, dar bir kenar çubuğunda **dikey**, geniş ana alanda **yatay** görünebilir — *ekran boyutundan bağımsız olarak*. Bileşen artık gerçekten **taşınabilir** hale gelir.

Duyarlı Tasarım Kontrol Listesi

Kontrol

- ☐ `<meta name = "viewport">` var mı? `user-scalable = no` yok değil mi?
- ☐ 320px genişlikte **yatay kaydırma** oluyor mu?
- ☐ `img { max-width: 100%; height: auto; }` tanımlı mı?
- ☐ Görsellere `width/height` yazılmış mı? (layout shift)
- ☐ Dokunma hedefleri en az **44×44px** mi?
- ☐ Gövde metni mobilde en az **16px** mi?
- ☐ Tablolar dar ekranda ne oluyor? (`overflow-x: auto` sarıcı)
- ☐ **%200 zoom**'da düzen bozuluyor mu?
- ☐ Media query'ler **mobile-first** (`min-width`) mi?
- ☐ Gereksiz media query var mı? (`auto-fit/clamp` ile çözülür mü?)
- ☐ **Gerçek bir telefonda** test edildi mi?

En sert test: DevTools cihaz modunda **320px** (iPhone SE) seçin. Burada çalışan tasarım, neredeyse her yerde çalışır.

Tablolar dar ekranlarda **en çok sorun çıkaran** öğedir. İçerik yapısal olduğu için kolayca “kırılmaz”.

✓ 1. Yatay kaydırma sarıcısı

```
<div class="tablo-sarici" tabindex="0"
  role="region" aria-label="Cok satanlar">
  <table> ... </table>
</div>
```

```
.tablo-sarici {
  overflow-x: auto;
  -webkit-overflow-scrolling: touch;
}
table { min-width: 520px; }
```

`tabindex = "0"` ile **klavyeyle de** kaydırılabilir olur.

2. Kart görünümüne dönüştürme

```
@media (max-width: 640px) {
  table, thead, tbody, tr, td { display: block; }
  thead { display: none; }
  tr { border: 1px solid var(--renk-cizgi);
    margin-bottom: 1rem; padding: 0.5rem; }
  td::before {
    content: attr(data-baslik) ": ";
    font-weight: 700;
  }
}
```

HTML’de `<td data-baslik = "Yazar">` yazmanız gerekir. **Dikkat:** Bu yöntem tablo semantiğini bozar — sadece basit tablolarda kullanın.

Öneri: Çoğu durumda **1. yöntem** (kaydırma sarıcısı) hem daha basit hem daha erişilebilirdir.

Sık Yapılan Duyarlılık Hataları

Hata	Sonuç ve Çözüm
Sabit <code>width: 1200px</code> kapsayıcı	Telefonda yatay kaydırma. <code>max-width</code> veya <code>width: min(1200px, 100% - 2rem)</code> .
Görsele <code>width</code> verip <code>max-width</code> vermemek	Görsel taşar. <code>max-width: 100%</code> ekleyin.
<code>100vw</code> kullanmak	Dikey kaydırma çubuğu genişliği dahil olur, yatay taşma yaratır. <code>100%</code> kullanın.
Mobilde <code>font-size: 14px</code>	iOS otomatik zoom yapar (input'ta). En az <code>16px</code> verin.
Küçük dokunma hedefi	Yanlış tıklama. <code>padding</code> ile <code>44×44px</code> 'e çıkarın.
<code>:hover</code> 'a bağımlı menü	Dokunmatikte çalışmaz. Tıklama tabanlı olmalı.
Sadece <code>max-width</code> media query	Desktop-first — fazla “geri alma” kodu. <code>min-width</code> ile mobile-first yazın.

En sinsi olanı: `100vw`. Masaüstü tarayıcıda kaydırma çubuğu yaklaşık **`15px`** yer kaplar ve `100vw` bunu saymaz — sayfa *tam* `15px` taşar.

Yapay Zeka ile İzgara ve Duyarlılık

BÖLÜM 3

Grid'de Yapay Zekanın Zayıf Noktaları



İyi

- ▶ `grid-template-areas` yazımı
- ▶ `auto-fit/minmax()` kalıbı
- ▶ Çizgi numarası hesabı
- ▶ Media query sözdizimi



Zayıf

- ▶ Gereksiz **media query** üretme — `auto-fit` varken 4 kırılma noktası yazar
- ▶ Desktop-first üretme eğilimi
- ▶ Eski `-ms-grid` örnekleri ekleme
- ▶ `px` tabanlı sabit ızgara önerme
- ▶ Cihaz boyutlarını ezbere kullanma



Dikkat

En sık göreceğiniz şey: Yapay zeka, `auto-fit + minmax` ile **tek satırda** çözülebilecek bir kart ızgarası için **dört ayrı media query** yazar. Prompt'unuza **"media query kullanma"** kısıtını ekleyin.

Yapay Zeka İstemi (Prompt) — Media Query'siz Izgara

Bir kitap sitesi için kart ızgarası CSS'i yaz.

GEREKSİNİMLER: - Kartlar en az 260px genişliğinde olsun - Sığan kadar sütun olussun, kalan boşluğu paylassınlar - Kartlar arası boşluk 1.5rem - Kart sayısı değiştiğinde düzen bozulmasın

KISITLAR (ÖNEMLİ): - @media KULLANMA - Sabit sütun sayısı (repeat(3, ...)) KULLANMA - float veya position KULLANMA - Toplam CSS 6 satırı geçmesin

Kodun altında, bu çözümün 1440px, 900px ve 375px genişliklerde KAC SÜTÜN üreteceğini hesapla ve göster.

Doğrulama: Hesabı DevTools cihaz modunda **gerçekten ölçün**. Tuttu mu?

Yapay Zeka İstemi (Prompt) — `grid-template-areas` ile İskelet

Bir kitap sitesi için MOBILE-FIRST sayfa iskeleti CSS'i yaz. `grid-template-areas` kullan.

Bolgeler: header, nav, main, aside, footer

DUZEN KURALLARI: - Mobil (varsayilan): hepsi tek sutun, sirasiyla header -> nav -> main

-> aside -> footer - Tablet ($> = 768\text{px}$): main ve aside yan yana (2fr / 1fr), header ve nav ustte tam genislik - Masaustu ($> = 1024\text{px}$): nav SOL tarafta dikey bir sutun, main ortada, aside sagda

KURALLAR: - Mobile-first yaz: temel stiller once, min-width ile ekle - HTML kaynak sirasini DEGISTIRME (erisilebilirlik) - Her @media blogunun ustune tek satirlik yorum koy - Sonunda: her breakpoint icin `grid-template-areas`'in gorsel semasini ASCII olarak ciz

Prompt 3 — Duyarlılık Denetimi

Yapay Zeka İstemi (Prompt) — Mobil Uyumluluk İncelemesi

Asagidaki CSS'i MOBİL UYUMLULUK acisindan denetle.

[CSS KODUNUZU YAPISTIRIN]

Su kontrolleri yap ve her biri icin [SORUN] veya [TAMAM] etiketi ver:

1) Sabit px genislik kullanilmis mi? (yatay kaydırma riski) 2) Media query'ler mobile-first mi, desktop-first mi? 3) Gereksiz media query var mi? (auto-fit/clamp ile cozulebilir mi?) 4) Dokunma hedefleri en az 44x44px mi? (buton, link) 5) Font boyutlari mobilde 16px altina dusuyor mu? 6) Gorseller icin max-width: 100% tanimli mi? 7) viewport meta etiketi user-scalable =no iceriyor mu?

Her [SORUN] icin duzeltilmis satiri "eski -> yeni" olarak goster. Tum dosyayi yeniden yazma.

Sonunda: kac tane media query'yi TAMAMEN kaldirabilecegimi ve yerine ne kullanacagimi belirt.

Yapay Zeka Çıktısını Doğrulama

4. maddeye dikkat: Yapay zeka dokunma hedefi boyutunu **hesaplayamaz** — **padding** ve **font-size**'dan tahmin eder. DevTools'ta elementi seçip gerçek boyutu ölçün.

Özet ve Değerlendirme

BÖLÜM 4

Grid

```
display: grid +  
grid-template-columns.  
fr = kalan boşluğun payı.  
repeat(auto-fit, minmax(260px,  
1fr)) — media query'siz duyarlı ızgara.
```

Alanlar

grid-template-areas ile düzenin **resmini çizim**. grid-column: 1 / -1 = baştan sona.

Duyarlılık

Viewport meta + akışkan düzen + media query.
Mobile-first: temel stiller mobil, `min-width` ile ekle.

Yapay Zeka

Gereksiz media query üretir. Prompt'a “**@media kullanma**” kısıtı ekleyin — çıktı dramatik biçimde kısalır.

Altın Kural: Önce **içsel olarak duyarlı** (`auto-fit`, `clamp`, `min`) çözümleri deneyin; media query'yi gerçekten gerektiğinde ekleyin.

Terim	Anlam	Terim	Anlam
Grid container	<code>display: grid</code> olan element	Breakpoint	Kırılma noktası
Grid item	Doğrudan çocuk	Mobile-first	Önce mobil, sonra ekle
fr	Kalan boşluğun payı	Viewport	Görüntülenen alan
repeat()	Tekrarlama fonksiyonu	srcset	Çoklu görsel kaynağı
minmax()	Alt/üst sınır	picture	Sanat yönetimi
auto-fit	Sığan kadar sütun	Layout shift	Yüklenirken zıplayan düzen
grid-area	Öğenin ızgaradaki alanı	clamp()	Min–tercih–max
place-items	Ortalama kısa yazımı	Intrinsic design	İçsel duyarlılık

- 1 Flexbox ile Grid arasındaki temel farkı tek cümleyle yazınız. Hangi durumda hangisini seçersiniz?
- 2 `grid-template-columns: 200px 2fr 1fr;` tanımlı 1000px genişliğindeki bir kapsayıcıda (gap yok) her sütun kaç piksel olur?
- 3 `repeat(auto-fit, minmax(260px, 1fr))` tanımı, 900px genişliğinde ve 20px gap ile kaç sütun üretir?
- 4 `auto-fit` ile `auto-fill` arasındaki fark nedir?
- 5 Viewport meta etiketi olmayan bir sayfada medya sorguları neden çalışmaz?
- 6 `user-scalable =no` yazmak neden bir erişilebilirlik ihlalidir?
- 7 Mobile-first yaklaşımın desktop-first'e göre üç avantajını yazınız.

Laboratuvar Föyü 9

- ▶ gridgarden.com 28 seviye
- ▶ `<main>` için iki sütunlu ızgara
- ▶ `auto-fit` kart ızgarası ve sütun sayısı
- ▶ `grid-template-areas` ile sayfa iskeleti
- ▶ Mobile-first dönüşüm
- ▶ Gerçek telefonda test
- ▶ Yapay zekaya kısıtlı ızgara üretirme

Hazırlık

Okuma: Kitap Ünite 2.2 — “Modern CSS Düzenleri (Grid)”, “Duyarlı Tasarım” **Kaynaklar:**

- ▶ css-tricks.com/snippets/css/complete-grid-primer/
- ▶ gridgarden.com
- ▶ gridbyexample.com

Gelecek hafta: **JavaScript başlıyor** — sayfamız artık düşünecek.

Ara değerlendirme: Bu hafta ile **HTML + CSS** bölümü tamamlandı. BookVerse'ün görsel tasarımı artık hazır.

Haftaya Görüşmek Üzere!

Gelecek Hafta:

JavaScript Temelleri

Değişkenler ▪ Veri tipleri ▪ Operatörler ▪ Koşullar ▪ Döngüler ▪ Konsol