WEB & YAZILIM GELİŞTİRME SERİSİ · MODÜL 3

HTML

Web'in iskeleti: etiketler, elementler ve öznitelikler; başlıklar, listeler, bağlantılar ve görseller; semantik yapı, tablolar, formlar ve erişilebilirlik. Gerçek kod ve özgün diyagramlarla, sıfırdan.

İlk gerçek kod modülü · Eğitim amaçlıdır

Bu Kitap Hakkında

Bu modül, web sayfalarının iskeletini kuran HTML'i sıfırdan öğretir. Dört seviye ve yirmi altı bölüm boyunca etiket–element–öznitelik mantığından semantik yapıya, tablolardan formlara ve erişilebilirlikten temel SEO'ya kadar her şeyi gerçek kod örnekleriyle ele alır.

Her bölümde çalışan kod blokları, konuyu görselleştiren özgün bir diyagram, 'tarayıcıda ne görünür?' kartı ve bir alıştırma yer alır. Bu, on altı modüllük 'Web & Yazılım Geliştirme' serisinin üçüncü ve ilk gerçek kod yazılan modülüdür; sıradaki modülde (CSS) bu iskelete görünüm kazandırırsın. Kod örneklerini kendi editöründe deneyerek ilerlemen önerilir. Bu seri eğitim amaçlıdır.

Web & Yazılım Geliştirme Serisi · Modül 3

İçindekiler

★ HTML Etiket Sözlüğü 61

SEVİYE 1

İlk Sayfa

HTML'in temelleri: HTML nedir, ilk sayfa iskeleti, etiket–element–öznitelik, başlıklar ve paragraflar, metin biçimlendirme, listeler, bağlantılar ve görseller.

BÖLÜM 01

HTML Nedir?

HTML (HyperText Markup Language), web sayfalarının iskeletidir. Etiketlerle içeriği işaretler: "bu bir başlık", "bu bir paragraf", "bu bir bağlantı". Tarayıcı bu işaretlemeyi okuyup sayfayı çizer. Web öğrenmenin ilk ve en doğal adımıdır.

Üçlü: yapı, görünüm, davranış

- **HTML:** sayfanın yapısı/iskeleti (başlık, paragraf, liste...).
- **CSS:** görünüm/stil (renk, yerleşim, yazı tipi).
- **JavaScript:** davranış/etkileşim (tıklama, hesaplama).

Şema 1.1 — Web'in üç temel teknolojisi; HTML iskeleti kurar.

HTML bir programlama dili mi?

- Hayır; HTML bir **işaretleme dilidir** — mantık veya karar içermez.
- Ama her web sayfasının temelidir ve öğrenmesi en kolay başlangıçtır.
- Birkaç saatte ilk sayfanı yapabilir, anında sonucu görebilirsin.

En küçük HTML parçası

```
<p>Merhaba, ben HTML.</p>
```

İPUCU

HTML'i bir **eve** benzet: HTML duvarlar ve odalar (yapı), CSS boya ve dekor (görünüm), JavaScript ise elektrik ve cihazlar (davranış). Önce sağlam bir yapı kurarsın; gerisi onun üstüne gelir. Bu yüzden HTML ile başlamak en mantıklısıdır.

Tarayıcıda ne görünür? ÖNİZLEME

`<p>Merhaba</p>` yazınca tarayıcı "Merhaba" metnini bir paragraf olarak gösterir. Etiketlerin kendisi (köşeli parantezler) görünmez; yalnızca işlevleri — yani metnin bir paragraf olduğu — görünür.

Alıştırma 8 dk

HTML'i tanı:

1. HTML, CSS ve JavaScript'in rollerini kendi cümlelerinle yaz.
2. Bir web sayfasında "yapı" sayılabilecek 3 öğe (başlık, liste...) say.
3. "İşaretleme dili" ne demek, açıkla.

BÖLÜM 02

İlk HTML Sayfası

Her HTML sayfası belirli bir iskeletle başlar. Bu iskeleti bir kez öğrenince, kuracağın her sayfada aynısını kullanırsın. Hadi ilk sayfanı oluşturalım.

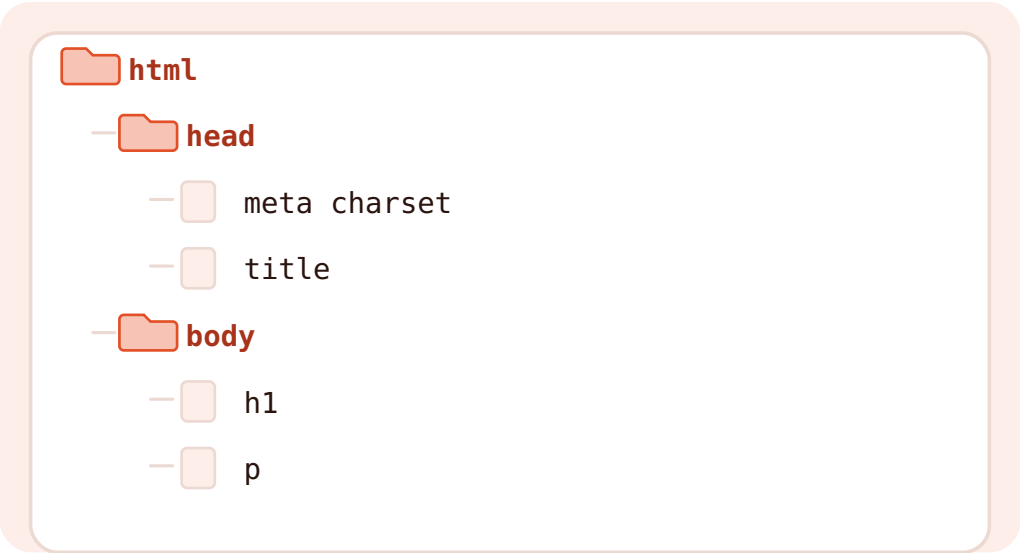
Temel iskelet

index.html — temel HTML iskeleti

```
<!doctype html>
<html lang="tr">
  <head>
    <meta charset="utf-8">
    <title>Sayfa Başlığı</title>
  </head>
  <body>
    <h1>Merhaba Dünya</h1>
    <p>İlk web sayfam.</p>
  </body>
</html>
```

Parçalar ne işe yarar?

- `<!doctype html>` — belge türünü (HTML5) bildirir.
- `<html>` — tüm sayfayı saran kök element.
- `<head>` — görünmeyen bilgiler (başlık, karakter seti).
- `<body>` — sayfada görünen içerik.



Şema 2.1 — Bir HTML belgesinin yapısı: head (görünmez bilgi) + body (görünen içerik).

İPUCU

`<head>` içindekiler sayfada **görünmez** (sekme başlığı, karakter seti gibi ayarlardır); gördüğün her şey `<body>` içindedir. Bu ayrımı erken oturt — yeni başlayanların en sık karıştırdığı şeydir.

Tarayıcıda ne görünür?

ÖNİZLEME

Bu iskeleti tarayıcıda açınca: sekmede "Sayfa Başlığı" yazar, sayfada büyük bir "Merhaba Dünya" başlığı ve altında "İlk web sayfam." paragrafı görünür.

Alıştırma

12 dk

İlk sayfanı yap:

1. Bir `index.html` dosyası oluştur.
2. Yukarıdaki iskeleti yaz (kendi başlığınla).
3. Dosyayı bir tarayıcıda açıp sonucu gör.

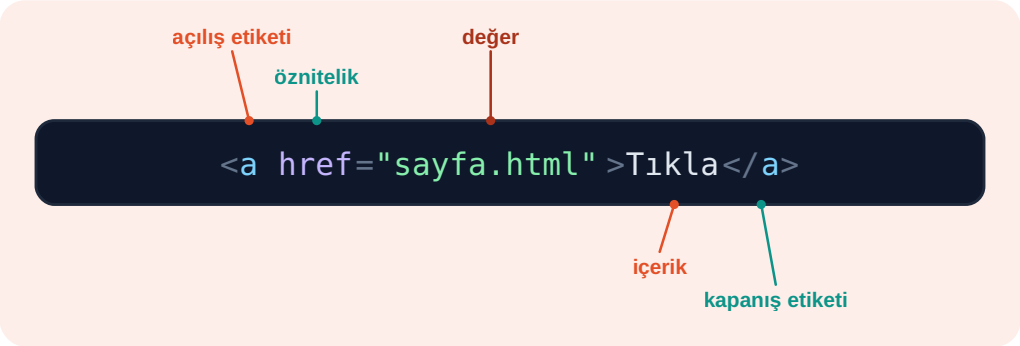
BÖLÜM 03

Etiketler, Elementler ve Öznitelikler

HTML üç temel kavram üstüne kuruludur: etiket, element ve öznitelik. Bunları anlamak, neredeyse tüm HTML'i anlamak demektir.

Tanımlar

- **Etiket (tag):** `<p>` gibi köşeli parantezli işaret (açılış ve kapanış).
- **Element:** açılış etiketi + içerik + kapanış etiketinin tamamı.
- **Öznitelik (attribute):** etikete ek bilgi veren değer (örn. `href`, `src`, `alt`).



Şema 3.1 — Bir HTML elementinin parçaları: açılış etiketi, öznitelik, değer, içerik ve kapanış etiketi.

Öznitelikli elementler

```
<a href="https://ornek.com">Tıkla</a>
<img src="logo.png" alt="Logo">
```

Açılış, kapanış ve "boş" elementler

- Çoğu element kapanış ister: `<p>...</p>`.
- Bazıları **boş (void)** elementtir, kapanışsızdır: `<img>`, `<br>`, `<hr>`.
- İç içe elementlerde sırayı koru: `<b><i>...</i></b>` (ters kapatma yok).

İPUCU

Kapanış etiketini unutmak en sık hatalardandır; VS Code çoğu zaman otomatik kapatır ve hatayı renkle gösterir. Öznitelik değerlerini her zaman **tırnak içinde** yaz: `href="..."`.

Tarayıcıda ne görünür? ÖNİZLEME

Öznitelikler tarayıcıda doğrudan görünmez ama davranışı belirler: `href` bağlantının nereye gideceğini, `alt` ise görsel yüklenmezse gösterilecek metni söyler.

Alıştırma

10 dk

Parçaları ayırt et:

1. Şema 3.1'deki parçaları (etiket, öznitelik, değer, içerik) kendi cümlelerinle açıkla.
2. Bir `<a>` ve bir `<img>` elementi yaz.
3. Hangisi "boş" element, işaretle.

BÖLÜM 04

Başlıklar ve Paragraflar

Metin, web'in büyük kısmını oluşturur. Başlıklar ve paragraflar, metni anlamlı biçimde düzenlemenin temel araçlarıdır.

Başlıklar: h1–h6

- `<h1>` en önemli başlık (sayfada genelde bir tane).
- `<h2>`'den `<h6>`'ya doğru alt başlıklar.
- Sıralı kullan (h1 → h2 → h3); seviye atlama.

Başlıklar ve paragraf

```
<h1>Ana Başlık</h1>
<h2>Alt Başlık</h2>
<p>Bu bir paragraf metnidir.</p>
<p>Bu ise ayrı bir paragraf.</p>
```

Şema 4.1 — Başlık seviyeleri tarayıcıda farklı boyutlarda; paragraflar normal metin olarak görünür.

Paragraflar

- `<p>...</p>` her paragrafı ayrı tutar.
- Tarayıcı paragraflar arasına otomatik boşluk koyar.

İPUCU

Başlık seviyesini **görünüm için değil anlam için** seç: bir metni "büyük görünsün" diye `<h1>` yapma; en önemli başlık o olduğu için yap. Boyutu sonra CSS ile ayarlarsın. Doğru başlık sırası, erişilebilirlik ve arama motorları için önemlidir.

Tarayıcıda ne görünür?

ÖNİZLEME

`<h1>` en büyük, `<h6>` en küçük görünür; `<p>` ise normal metin olarak, paragraflar arasında boşlukla gösterilir.

Alıştırma

8 dk

Metni düzenle:

1. Bir başlık ve iki paragraftan oluşan bir bölüm yaz.
2. Bir alt başlık (`h2`) ekle.
3. Tarayıcıda boyut farkını gözlemle.

BÖLÜM 05

Metin Biçimlendirme

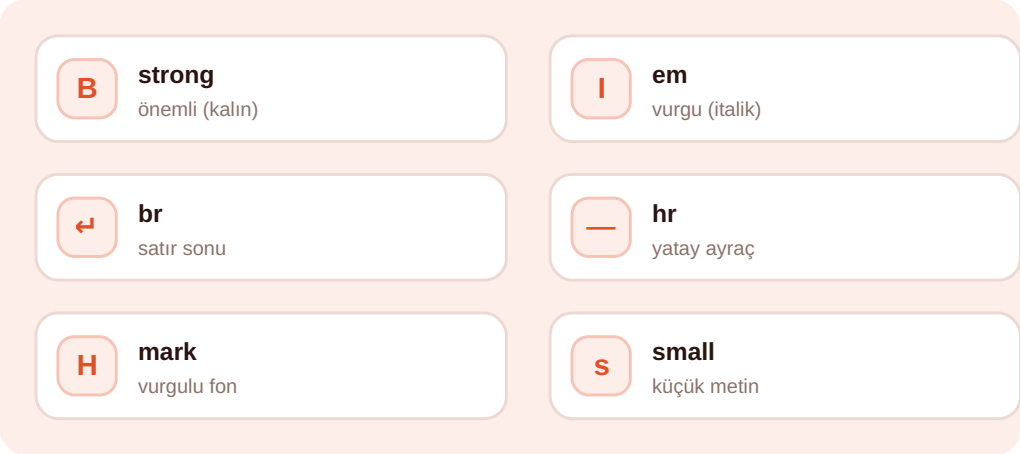
Metnin bazı kısımlarını vurgulamak veya araya yapı eklemek için satır içi (inline) etiketler kullanılır.

Vurgu ve satır etiketleri

- `<strong>` — önemli (kalın görünür).
- `<em>` — vurgu (italik görünür).
- `<br>` — satır sonu. `<hr>` — yatay ayraç.

Satır içi biçimlendirme

```
<p><strong>Önemli</strong> ve <em>vurgulu</em> metin.</p>
<p>İlk satır<br>İkinci satır</p>
<hr>
```



Şema 5.1 — Sık kullanılan metin biçimlendirme etiketleri.

İPUCU

`<strong>` ve `<em>` sadece görünüm (kalın/italik) değil, **anlam** taşır (önem/vurgu); ekran okuyucular bunu sesle yansıtır. Sadece görsel kalınlık istiyorsan bunu CSS ile yap; anlam için ise bu etiketler doğru seçimdir.

Tarayıcıda ne görünür? ÖNİZLEME

`<strong>` kalın, `<em>` italik görünür; `<br>` metni alt satıra geçirir, `<hr>` yatay bir çizgi çeker.

Alıştırma

8 dk

Metni biçimle:

1. Bir paragrafta bir kelimeyi `strong`, birini `em` yap.
2. İki satırı `br` ile ayır.
3. Bir `hr` ekleyip sonucu gör.

BÖLÜM 06

Listeler

Listeler, sıralı veya sırasız öğeleri düzenlemenin temel yoludur; menülerden adım adım yönergelere kadar her yerde karşına çıkar.

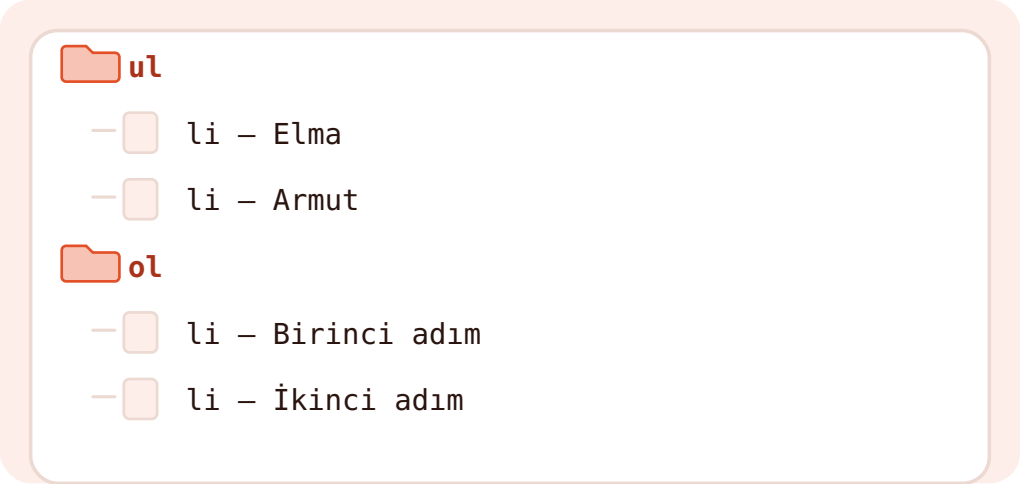
İki tür liste

- `<ul>` — sırasız liste (madde işareti).
- `<ol>` — sıralı liste (numara).
- `<li>` — her bir liste öğesi.

Sırasız ve sıralı liste

```
<ul>
  <li>Elma</li>
  <li>Armut</li>
</ul>

<ol>
  <li>Birinci adım</li>
  <li>İkinci adım</li>
</ol>
```



Şema 6.1 — Liste yapısı: ul/ol kapsayıcısı içinde li öğeleri.

İç içe listeler

- Bir `<li>` içine başka bir `<ul>` / `<ol>` koyarak alt listeler oluşturabilirsin.
- Navigasyon menüleri genelde `<ul>` ile yapılır.

İPUCU

Liste yalnızca "madde" göstermek için değil, **yapısal gruplama** için de kullanılır; menüler, adım listeleri, bağlantı grupları hep listelerle yapılır. İç içe listelerde her `<ul>` / `<ol>`'nin kapanışına dikkat et.

Tarayıcıda ne görünür?

ÖNİZLEME

`<ul>` madde işaretli, `<ol>` numaralı olarak görünür; her `<li>` ayrı bir satırda listelenir.

Alıştırma

8 dk

Liste yap:

1. Sevdiğin 3 şeyi bir `ul` ile listele.
2. Bir tarifi/adımları `ol` ile sırala.
3. Bir `li` içine küçük bir alt liste ekle.

BÖLÜM 07

Bağlantılar (Links)

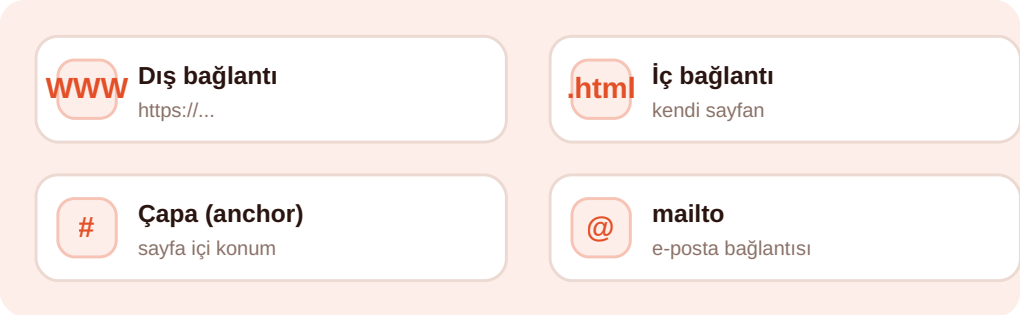
Bağlantılar, web'i "web" yapan şeydir; sayfaları ve siteleri birbirine bağlar. `<a>` (anchor) etiketiyle oluşturulur.

Bağlantı türleri

- **Dış bağlantı:** başka bir siteye — `href="https://..."`.
- **İç bağlantı:** kendi sayfalarına — `href="hakkinda.html"`.
- **Sayfa içi (çapa):** aynı sayfada bir yere — `href="#bolum2"`.

Farklı bağlantı türleri

```
 <a href="https://ornek.com">Dış site</a>
<a href="iletisim.html">İletişim sayfası</a>
<a href="#altbilgi">Sayfada aşağı git</a>
<a href="mailto:bilgi@ornek.com">E-posta gönder</a>
```



Şema 7.1 — Bağlantı türleri.

Yeni sekmede açma

- `target="_blank"` bağlantıyı yeni sekmede açar.
- Güvenlik için `rel="noopener"` eklemek iyi bir alışkanlıktır.

İPUCU

Bağlantı metni **anlamlı** olsun: "buraya tıkla" yerine "İletişim sayfası" yaz. Açıklayıcı bağlantı metni hem kullanıcı hem de ekran okuyucu için çok daha iyidir; "tıkla" tek başına hiçbir şey anlatmaz.

Tarayıcıda ne görünür? ÖNİZLEME

Bağlantılar genelde altı çizili ve farklı renkte görünür; tıklanınca `href`'te belirtilen adrese gidilir. `mailto` bağlantısı e-posta uygulamasını açar.

Alıştırma

10 dk

Bağlantı kur:

1. Bir dış siteye ve bir iç sayfaya bağlantı yaz.
2. Sayfa içi bir çapa bağlantısı (#) oluştur.
3. Bir bağlantıya anlamlı bir metin ver (asla "tıkla" değil).

BÖLÜM 08

Görseller

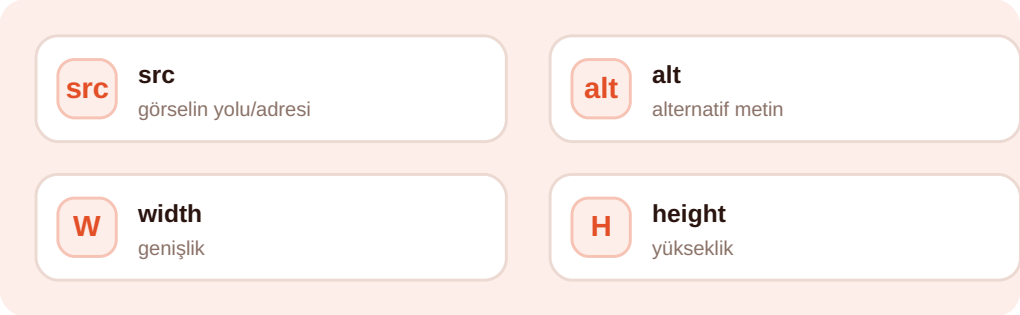
Görseller sayfaya hayat katar. `<img>` etiketiyle eklenir ve doğru kullanım için birkaç önemli özniteliği vardır.

Temel kullanım

- `src` — görselin yolu veya adresi.
- `alt` — görsel yüklenmezse / ekran okuyucu için açıklama.
- `width` / `height` — boyut (genelde CSS tercih edilir).

Görsel ekleme

```
<img src="resimler/logo.png" alt="Şirket logosu">
<img src="kedi.jpg" alt="Uyuyan bir kedi" width="300">
```



Şema 8.1 — etiketinin temel öznitelikleri.

alt neden önemli?

- Görsel yüklenmezse `alt` metni o alanda görünür.
- Görme engelliler için ekran okuyucu `alt`'ı okur.
- Arama motorları görseli `alt` ile anlar (SEO).

İPUCU

Her anlamlı görsele **alt** yaz; yalnızca süs olan görsellerde boş bırakılabilir (`alt=""`). İyi bir alt metni, görseli göremeyenler için sayfayı erişilebilir kılar — bu hem etik hem birçok yerde yasal bir gerekliliktir.

Tarayıcıda ne görünür? ÖNİZLEME

`<img>` görseli sayfada gösterir; görsel yüklenmezse `alt` metni o alanda görünür. `width` ile genişliği ayarlanır.

Alıştırma 10 dk

Görsel ekle:

1. Bir görseli `src` ve anlamlı bir `alt` ile ekle.
2. `src`'yi bilerek yanlış yazıp `alt`'ın göründüğünü gör.
3. Bir görsele `width` ver.

SEVİYE 2

Yapı ve İçerik

Sayfayı anlamlı kurmak: semantik HTML, bölümleme ve düzen (div/span), tablolar, formların temelleri, form öğeleri ve HTML ile form doğrulama.

BÖLÜM 09

Semantik HTML

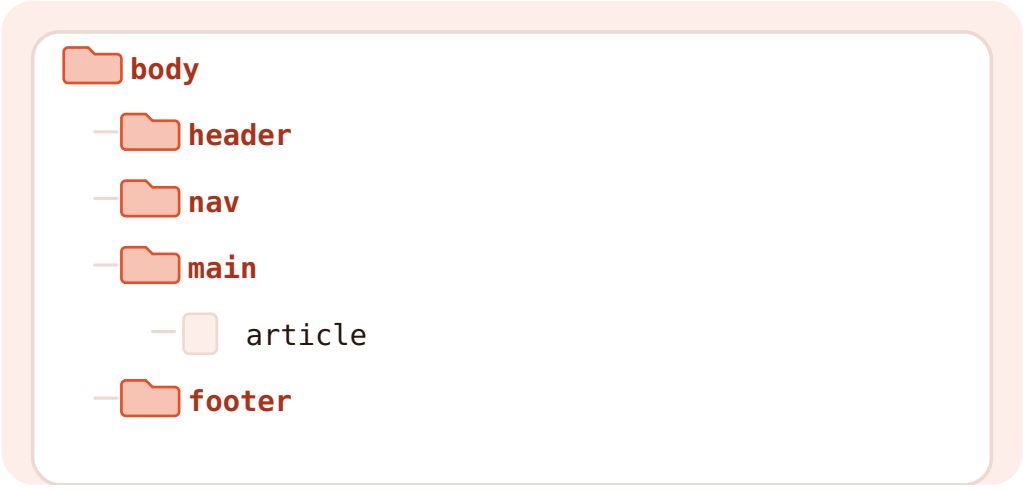
Semantik etiketler, içeriğin "ne olduğunu" anlam olarak söyler: bu bir başlık alanı, bu menü, bu ana içerik, bu altbilgi. Tarayıcılar, arama motorları ve ekran okuyucular bu anlamı kullanır.

Başlıca semantik etiketler

- `<header>` — üst alan (logo, başlık).
- `<nav>` — gezinme menüsü.
- `<main>` — sayfanın ana içeriği (sayfada bir tane).
- `<section>` / `<article>` — bölümler / bağımsız içerik.
- `<footer>` — altbilgi.

Semantik bir sayfa iskeleti

```
<body>
  <header>Logo ve başlık</header>
  <nav>Menü</nav>
  <main>
    <article>Ana yazı</article>
  </main>
  <footer>İletişim · Telif</footer>
</body>
```



Şema 9.1 — Semantik etiketlerle sayfanın bölgeleri anlam kazanır.

İPUCU

Aynı görünümü `<div>` ile de elde edebilirsin; ama semantik etiketler **anlam** katar. Ekran okuyucu "ana içeriğe geç" diyebilir, arama motoru sayfanı daha iyi anlar. Görünüm aynı olsa bile semantik HTML daha kalitelidir.

Tarayıcıda ne görünür? ÖNİZLEME

Semantik etiketler görünümü kendiliğinden çok değiştirmez (onu CSS yapar); ama sayfanın bölgelerini tarayıcıya, arama motoruna ve ekran okuyucuya anlamlı biçimde bildirir.

Alıştırma 10 dk

Semantik kur:

1. Bir sayfayı header, nav, main, footer ile yapılandır.
2. Ana içeriği `<main>` içine koy.
3. Neden `<div>` yerine semantik etiket tercih edilir, yaz.

BÖLÜM 10

Bölümleme ve Düzen: div ve span

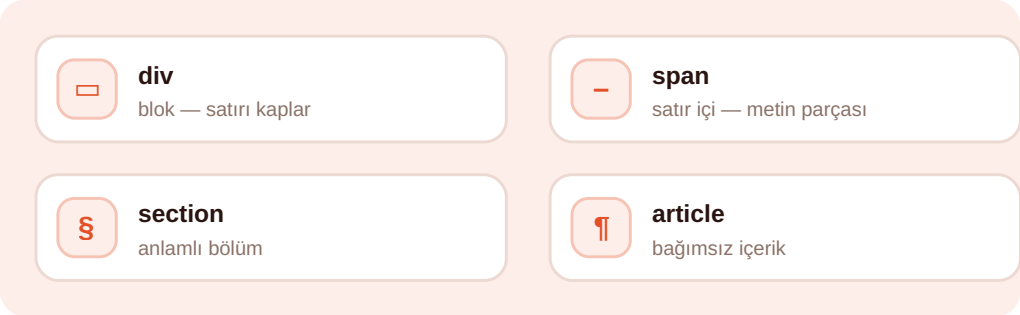
Bazen yalnızca gruplamak için anlamı olmayan genel kapsayıcılar gerekir. `<div>` ve `<span>` tam da bunun içindir; özellikle CSS ile düzen kurarken çok kullanılır.

Blok ve satır içi

- `<div>` — **blok** kapsayıcı; tüm satırı kaplar, büyük bölümleri gruplar.
- `<span>` — **satır içi** kapsayıcı; metnin küçük bir parçasını sarar.
- İkisinin de kendiliğinden bir anlamı/görünümü yoktur; CSS ile şekillenir.

div ve span

```
<div class="kart">
  <p>Fiyat: <span class="tutar">100 TL</span></p>
</div>
```



Şema 10.1 — Genel kapsayıcılar (div/span) ve semantik alternatifleri.

İPUCU

Önce semantik etiketi düşün; uygun semantik etiket yoksa `<div>` / `<span>` kullan. Her şeyi `<div>` ile yapmak ("div çorbası") çalışır ama kaliteyi düşürür. `class` öznitelikleri ileride CSS ile stillemek içindir.

Tarayıcıda ne görünür? ÖNİZLEME

`<div>` kendi başına görünür bir şey üretmez; bir kutu/grup oluşturur. `<span>` ise metnin içinde, akışı bozmadan küçük bir parçayı işaretler. Görünümü CSS belirler.

Alıştırma

8 dk

Kapsayıcıları kullan:

1. Bir `<div>` ile bir "kart" grupla.
2. Metin içinde bir kelimeyi `<span>` ile sar.
3. Hangisi blok, hangisi satır içi, işaretle.

BÖLÜM 11

Tablolar

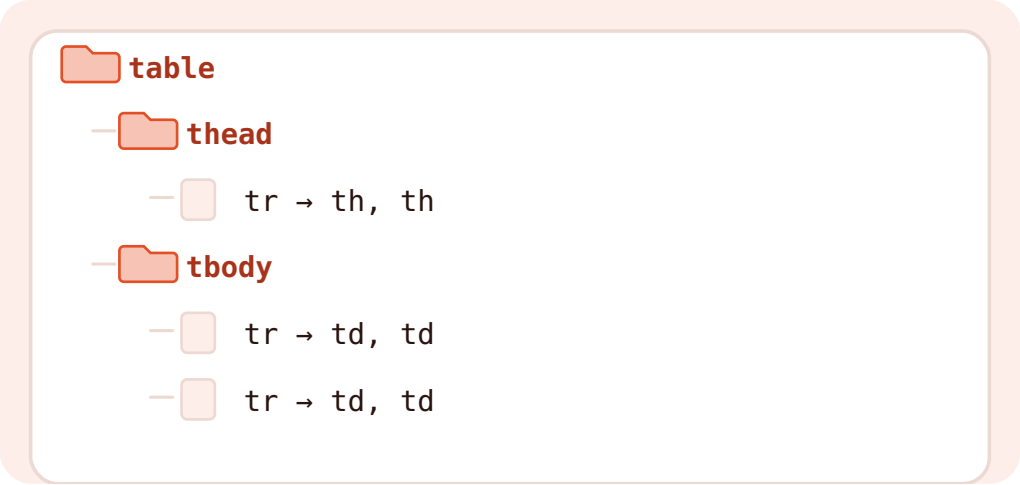
Tablolar, satır ve sütunlardan oluşan verileri (fiyat listesi, program, karşılaştırma) göstermek içindir. Düzen kurmak için değil, gerçek tablo verisi için kullanılır.

Tablo yapı taşları

- `<table>` — tabloyu sarar.
- `<tr>` — satır (table row).
- `<th>` — başlık hücresi. `<td>` — veri hücresi.
- `<thead>` / `<tbody>` — başlık ve gövde grupları.

Basit bir tablo

```
<table>
  <thead>
    <tr><th>Ürün</th><th>Fiyat</th></tr>
  </thead>
  <tbody>
    <tr><td>Kalem</td><td>10 TL</td></tr>
    <tr><td>Defter</td><td>25 TL</td></tr>
  </tbody>
</table>
```



Şema 11.1 — Tablo yapısı: satırlar (tr) içinde başlık (th) ve veri (td) hücreleri.

İPUCU

Tabloyu **sadece tablo verisi** için kullan; sayfa düzeni (yan yana kutular) kurmak için değil. Düzeni CSS ile yaparsın. Eskiden tablolarla düzen kurulurdu; bugün bu yanlış bir alışkanlıktır.

Tarayıcıda ne görünür? ÖNİZLEME

Tablo, satır ve sütunlardan oluşan bir ızgara olarak görünür; `<th>` hücreleri genelde kalın ve ortalı (başlık), `<td>` hücreleri normal görünür.

Alıştırma 10 dk

Tablo yap:

1. İki sütunlu (Ürün, Fiyat) bir tablo oluştur.
2. Başlık satırını `<th>` ile yap.
3. En az üç veri satırı ekle.

BÖLÜM 12

Formlar: Temeller

Formlar, kullanıcıdan veri almanın yoludur: giriş, kayıt, arama, iletişim. `<form>` etiketi ve içindeki giriş alanlarıyla oluşturulur.

Temel parçalar

- `<form>` — tüm form alanlarını sarar.
- `<label>` — bir alanın etiketi/açıklaması.
- `<input>` — metin, e-posta gibi giriş alanı.
- `<button>` — gönder düğmesi.

Basit bir form

```
<form>
  <label for="ad">Adınız</label>
  <input id="ad" type="text" name="ad">
  <button type="submit">Gönder</button>
</form>
```

Şema 12.1 — Bir formun yaşam döngüsü: göster → doldur → gönder → işle.

İPUCU

Her giriş alanını bir `<label>` ile eşleştir (`for` = input'un `id` 'si). Bu hem erişilebilirlik için şarttır (ekran okuyucu alanı tanır) hem de kullanıcı etikete tıklayınca alan seçilir. Etiketsiz form alanı kötü bir uygulamadır.

Tarayıcıda ne görünür? ÖNİZLEME

Form, ekranda giriş kutuları ve bir gönder düğmesi olarak görünür; `<label>` alanın yanında açıklama metni olarak çıkar. Düğmeye basınca form gönderilir.

Alıştırma 10 dk

Form kur:

1. Ad ve e-posta alanı olan bir form yap.
2. Her alana `<label>` ekle (`for` / `id` eşleşsin).
3. Bir gönder düğmesi ekle.

BÖLÜM 13

Form Öğeleri

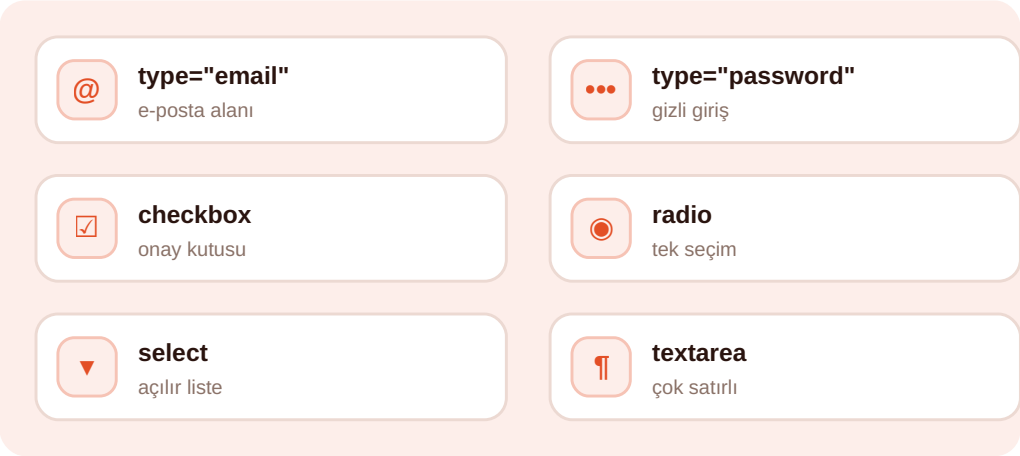
Formlar yalnızca metin kutusundan ibaret değildir. Farklı veri türleri için farklı giriş öğeleri vardır: e-posta, şifre, onay kutusu, seçenek, açılır liste ve daha fazlası.

Sık kullanılan öğeler

- `<input type="email/password/number/date">` — farklı türde alanlar.
- `<textarea>` — çok satırlı metin.
- `<select>` + `<option>` — açılır liste.
- `type="checkbox"` / `type="radio"` — onay kutusu / tek seçim.

Çeşitli form öğeleri

```
 <input type="email" placeholder="ornek@site.com">
<textarea rows="4"></textarea>
<select>
  <option>İstanbul</option>
  <option>Ankara</option>
</select>
<input type="checkbox" id="kvkk"> <label for="kvkk">Onaylıyorum</label>
```



Şema 13.1 — Farklı veri türleri için farklı form öğeleri.

İPUCU

Doğru `type`'ı seçmek önemlidir: `type="email"` telefonda e-posta klavyesi açar ve temel doğrulama yapar; `type="date"` tarih seçici gösterir. Doğru tür, hem kullanıcı deneyimini hem doğrulamayı iyileştirir.

Tarayıcıda ne görünür? ÖNİZLEME

Her öğe kendi görünümüyle çıkar: onay kutusu kare, radio yuvarlak, select açılır ok, textarea büyük bir kutu. `placeholder` kutuda soluk bir ipucu metni gösterir.

Alıştırma 10 dk

Öğeleri dene:

1. Bir e-posta ve bir şifre alanı ekle (doğru `type` ile).
2. Bir açılır liste (`select`) oluştur.
3. Bir onay kutusu ve etiketini ekle.

BÖLÜM 14

Form Doğrulama (HTML)

Kullanıcı hatalı veya eksik veri gönderebilir. HTML, JavaScript yazmadan bile temel doğrulama yapabilir: zorunlu alan, doğru biçim, en az/çok karakter gibi.

Yerleşik doğrulama öznitelikleri

- `required` — alan boş bırakılamaz.
- `type="email"` — geçerli e-posta biçimi ister.
- `minlength` / `maxlength` — karakter sınırı.
- `pattern` — belirli bir desen (örn. sadece rakam).

Yerleşik doğrulamalı form

```
 <input type="email" required>
<input type="text" minlength="3" required>
<input type="text" pattern="[0-9]{5}"
       title="5 haneli posta kodu">
```

Şema 14.1 — HTML doğrulaması: gönderimden önce tarayıcı alanları kontrol eder.

İPUCU

HTML doğrulaması **ilk savunma hattıdır** ama tek başına yeterli değildir; gerçek güvenlik için sunucu tarafında da doğrulama gerekir (ileri modüllerde). Yine de yerleşik doğrulama, kullanıcıya anında geri bildirim verdiği için çok değerlidir.

Tarayıcıda ne görünür? ÖNİZLEME

`required` bir alanı boş gönderirsen tarayıcı "bu alanı doldurun" uyarısı gösterir ve gönderimi durdurur. `type="email"` hatalı biçimde "geçerli bir e-posta girin" der.

Alıştırma 10 dk

Doğrulama ekle:

1. Bir alana `required` ekleyip boş göndermeyi dene.
2. Bir `type="email"` alanına hatalı metin gir.
3. `minlength` ile en az karakter kuralı koy.

SEVİYE 3

Profesyonel HTML

Kaliteyi artırmak: erişilebilirlik, meta etiketler ve head, multimedya, karakter varlıkları, HTML'in CSS/JS ile bağlantısı ve tam bir sayfa iskeleti.

BÖLÜM 15

Erişilebilirlik (a11y)

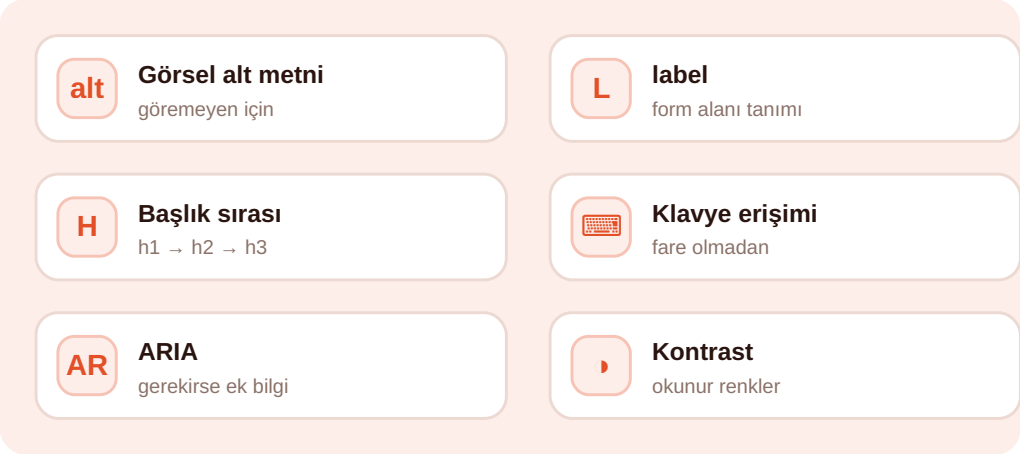
Erişilebilirlik, sayfanın herkes için — görme/işitme/hareket engelliler dahil — kullanılabilir olmasıdır. İyi HTML, erişilebilirliğin temelidir ve çoğu zaman zaten doğru etiketleri kullanmakla başlar.

Temel erişilebilirlik kuralları

- Görsellere anlamlı `alt` yaz.
- Form alanlarını `<label>` ile eşleştir.
- Başlık sırasını koru (h1 → h2 → h3).
- Semantik etiketler kullan (nav, main, button...).
- Klavyeyle gezilebilir olsun; renk tek başına anlam taşımasın.

Erişilebilir işaretleme

```
<img src="grafik.png" alt="2024 satışları %20 arttı">
<label for="tel">Telefon</label>
<input id="tel" type="tel">
<button aria-label="Menüyü aç">☰</button>
```



Şema 15.1 — Erişilebilirliğin temel taşları.

İPUCU

Erişilebilirlik sonradan eklenen bir "ekstra" değil; **baştan doğru HTML yazmakla** büyük ölçüde sağlanır. Semantik etiketler, alt metinleri ve label'lar zaten doğru kullanıldığında sayfan çoğu kişi için erişilebilir olur. ARIA'yı ancak HTML yetmediğinde ekle.

Tarayıcıda ne görünür? ÖNİZLEME

Erişilebilirlik çoğu zaman görünümü değiştirmez; ama ekran okuyucu kullanan biri için sayfayı kullanılabilir kılar: görselleri "duyar", form alanlarını tanır, bölümler arasında gezebilir.

Alıştırma 10 dk

Erişilebilir yap:

1. Bir görsele bilgi veren bir `alt` yaz (sadece "resim" değil).
2. Bir form alanını `<label>` ile eşleştir.
3. Sadece ikon olan bir düğmeye `aria-label` ekle.

BÖLÜM 16

Meta Etiketler ve

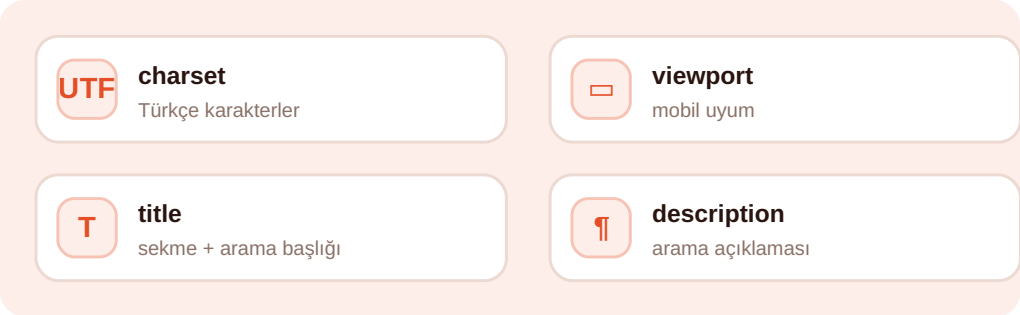
`<head>` sayfada görünmez ama çok önemlidir: karakter seti, mobil uyum, sayfa başlığı ve arama motorlarına/sosyal medyaya verilen bilgiler buradadır.

Olmazsa olmaz head öğeleri

- `<meta charset="utf-8">` — Türkçe karakterler düzgün görünür.
- `<meta name="viewport" ...>` — mobil uyum için şart.
- `<title>` — sekme ve arama sonucundaki başlık.
- `<meta name="description">` — arama sonucundaki açıklama.

Sağlam bir

```
<head>
  <meta charset="utf-8">
  <meta name="viewport"
        content="width=device-width, initial-scale=1">
  <title>Pastane — Taze Ekmek</title>
  <meta name="description" content="Her gün taze ekmek.">
</head>
```



Şema 16.1 — içindeki temel meta bilgiler.

İPUCU

`viewport` meta etiketini unutma; o olmadan sayfan telefonda küçücük ve okunmaz görünür. `charset="utf-8"` ise Türkçe karakterlerin (ç, ş, ğ, ı) bozulmaması için şarttır. Bu ikisi her sayfada olmalı.

Tarayıcıda ne görünür? ÖNİZLEME

`<head>` içeriği sayfada görünmez; ama `<title>` sekmede ve Google sonucunda başlık olarak, `description` ise o sonucun altındaki açıklama olarak çıkar.

Alıştırma

8 dk

Head'i tamamla:

1. Sayfana `charset` ve `viewport` ekle.
2. Anlamlı bir `<title>` yaz.
3. Bir `meta description` ekle.

BÖLÜM 17

Multimedya: Ses, Video, Gömme

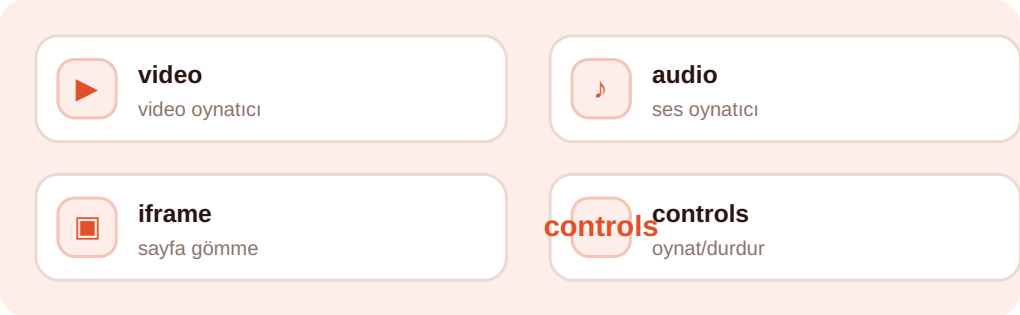
HTML sadece metin ve görsel değil; ses, video ve başka sayfaları gömmeyi de destekler. Bunlar modern web deneyiminin parçasıdır.

Multimedya etiketleri

- `<video>` — video oynatıcı (controls ile kontroller).
- `<audio>` — ses oynatıcı.
- `<iframe>` — başka bir sayfayı/haritayı/videoyu gömme.

Video ve gömme

```
<video src="tanitim.mp4" controls width="400"></video>
<audio src="muzik.mp3" controls></audio>
<iframe src="https://harita.ornek.com"
        title="Konum haritası"></iframe>
```



Şema 17.1 — Multimedya ve gömme etiketleri.

İPUCU

`<video>` ve `<audio>`'ya `controls` eklemeyi unutma; yoksa kullanıcı oynatamaz. `<iframe>` güçlüdür ama güvenlik açısından dikkatli kullan: yalnızca güvendiğin kaynakları göm.

Tarayıcıda ne görünür? ÖNİZLEME

`<video controls>` ekranda oynat/durdur düğmeli bir oynatıcı gösterir; `<iframe>` ise belirtilen sayfayı küçük bir pencere gibi sayfanın içine gömer.

Alıştırma

8 dk

Multimedya ekle:

1. Bir `<video>` veya `<audio>` ekle (`controls` ile).
2. Bir `<iframe>` ile bir sayfa göm.
3. `iframe` 'e `title` ekle (erişilebilirlik).

BÖLÜM 18

Karakter Varlıkları ve Özel Karakterler

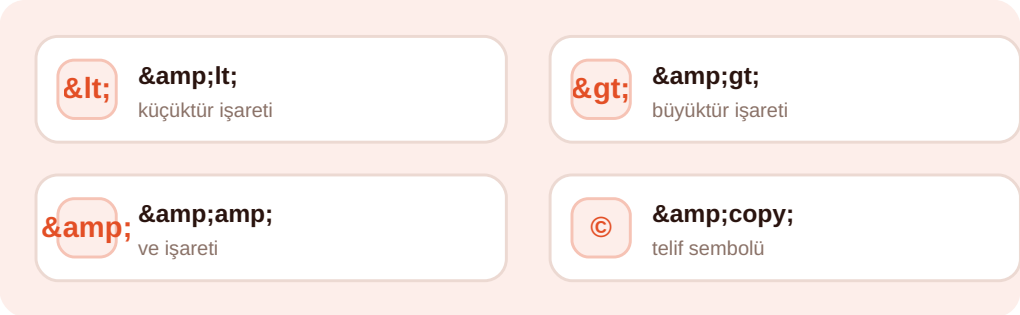
Bazı karakterlerin HTML'de özel anlamı vardır (örn. < ve >). Onları metin olarak göstermek veya özel sembolleri eklemek için "karakter varlıkları" (entities) kullanılır.

Sık kullanılan varlıklar

- `<` → `<` ve `>` → `>`
- `&` → `&` (ve işareti)
- `&copy;` → © , ` ` → boşluk

Özel karakterleri gösterme

```
<p>5 &lt; 10 ve 10 &gt; 5</p>
<p>Ahmet &amp; Mehmet</p>
<p>&copy; 2025 Şirket</p>
```



Şema 18.1 — Sık kullanılan karakter varlıkları.

İPUCU

Türkçe karakterler (ç, ş, ğ, ı, ö, ü) için varlık kullanman **gerekmez**; `<meta charset="utf-8">` varsa doğrudan yazabilirsin. Varlıklar asıl `<`, `>`, `&` gibi HTML'de özel anlamı olan karakterler için gerekir.

Tarayıcıda ne görünür? ÖNİZLEME

`<` yazınca tarayıcı ekranda `<` karakterini gösterir (etiket başlangıcı olarak yorumlamadan). Böylece kod örnekleri veya matematik işaretleri metin olarak görünebilir.

Alıştırma

8 dk

Varlıkları kullan:

1. Bir paragrafta `<` ve `>` karakterlerini metin olarak göster.
2. Bir telif (©) sembolü ekle.
3. Türkçe karakterlerin `charset` ile düzgün geldiğini doğrula.

BÖLÜM 19

HTML'in CSS ve JS ile Bağlantısı

HTML tek başına yapı kurar; ama güzel görünüm için CSS'i, etkileşim için JavaScript'i bağlaman gerekir. Bu bağlantı birkaç etiketle yapılır ve sonraki modüllerin köprüsüdür.

Bağlama etiketleri

- `<link rel="stylesheet" href="styles.css">` — CSS dosyasını bağlar (head içinde).
- `<script src="app.js"></script>` — JavaScript dosyasını bağlar (genelde body sonunda).
- Böylece yapı (HTML), görünüm (CSS) ve davranış (JS) ayrı dosyalarda, düzenli durur.

CSS ve JS'i bağlama

```
<head>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  ... içerik ...
  <script src="app.js"></script>
</body>
```

Şema 19.1 — HTML, link ile CSS'i ve script ile JS'i kendine bağlar.

İPUCU

Üç teknolojiyi **ayrı dosyalarda** tutmak iyi bir alışkanlıktır: `index.html`, `styles.css`, `app.js`. Bu, kodu düzenli ve bakımı kolay tutar. `<script>` etiketini genelde `<body>` sonuna koyarsın ki sayfa önce görünsün.

Tarayıcıda ne görünür? ÖNİZLEME

`<link>` eklenince sayfa CSS'teki stillerle (renk, düzen) görünmeye başlar; `<script>` eklenince JavaScript çalışır ve sayfa etkileşimli (tıklanınca tepki veren) hâle gelir.

Alıştırma 8 dk

Bağlantıyı kur:

1. Bir `styles.css` dosyasını `<link>` ile bağla.
2. Bir `app.js` dosyasını `<script>` ile bağla.
3. Üçünü ayrı dosyalarda tutmanın faydasını yaz.

BÖLÜM 20

Tam Bir Sayfa İskeleti Kurmak

Şimdiye kadar öğrendiklerini birleştirip gerçekçi, tam bir sayfa iskeleti kuruyorsun: head, semantik bölgeler ve içerik bir arada.

Bir araya getirmek

Tam bir sayfa iskeleti

```
<!doctype html>
<html lang="tr">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Pastane</title>
</head>
<body>
  <header><h1>Taze Pastane</h1></header>
  <nav><a href="#urunler">Ürünler</a></nav>
  <main>
    <section id="urunler"><h2>Ürünler</h2></section>
  </main>
  <footer>&copy; 2025</footer>
</body>
</html>
```

Şema 20.1 — Tam bir sayfanın yapısı: head + semantik body bölgeleri.

İPUCU

Bu iskeleti bir **şablon** olarak sakla; her yeni sayfaya buradan başla. Profesyoneller her projeye sıfırdan değil, denenmiş bir iskeletten (boilerplate) başlar. Zamanla bu senin standart başlangıcın olur.

Tarayıcıda ne görünür?

ÖNİZLEME

Bu iskelet tarayıcıda: üstte başlık (header), altında menü (nav), ortada ürünler bölümü (main/section) ve en altta telifli bir altbilgi (footer) olarak görünür — hepsi anlamlı biçimde sıralı.

Alıştırma

12 dk

Tam sayfa kur:

1. Yukarıdaki gibi tam bir sayfa iskeleti yaz.
2. Kendi konunla (kafe, blog, portfolyo) doldur.
3. Tarayıcıda açıp bölgelerin yerini gözlemle.

SEVİYE 4

Sağlam Sayfalar

Ustalık: anlamlı ve erişilebilir yapı, SEO ve paylaşım etiketleri, performans ve temiz kod, şablonlar, HTML5 API'lerine bakış ve çok sayfalı bir site iskeleti.

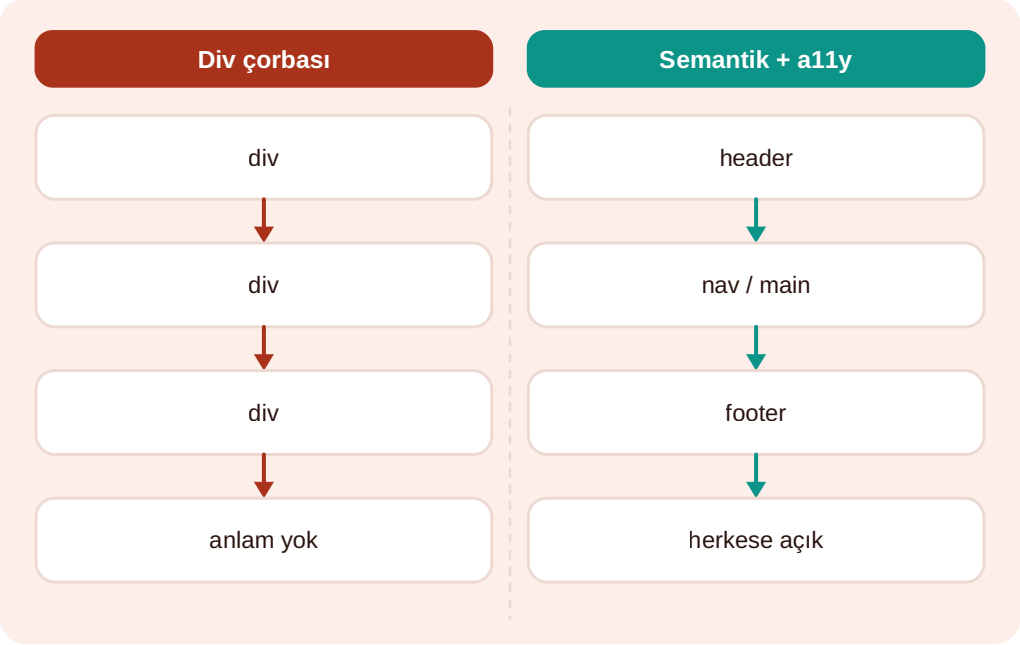
BÖLÜM 21

Anlamlı ve Erişilebilir Yapı

İyi HTML, hem makineler (arama motoru, ekran okuyucu) hem insanlar için anlamlıdır. "div çorbası" yerine semantik ve erişilebilir bir yapı, profesyonelliğin işaretidir.

İki yaklaşım

- **Sadece div:** her şey anlamsız `<div>`; çalışır ama makineler anlamaz.
- **Semantik + erişilebilir:** doğru etiketler; herkes için kullanılabilir.
- Görünüm aynı olabilir, ama kalite ve erişilebilirlik çok farklıdır.



Şema 21.1 — Aynı görünüm, çok farklı kalite: anlamsız div yığını vs semantik yapı.

Anlamlı yapı

```
 <!-- iyi -->
<nav><ul><li><a href="/">Ana sayfa</a></li></ul></nav>
<!-- zayıf -->
<div><div><div>Ana sayfa</div></div></div>
```

İPUCU

Bir element yazmadan önce sor: "Bunun anlamı ne?" Eğer bir başlık ise `<h2>`, bir gezinme ise `<nav>`, bir düğme ise `<button>` kullan. Doğru element, hem erişilebilirliği hem SEO'yu bedavaya getirir.

Tarayıcıda ne görünür?

ÖNİZLEME

İki yaklaşım da ekranda aynı görünebilir; ama ekran okuyucu kullanan biri için semantik yapı "gezinme menüsü", "ana içerik" gibi ipuçları verirken, div yığını sadece anlamsız metin yığınıdır.

Alıştırma

10 dk

Anlamlı yaz:

1. Bir "div çorbası" bölümü, semantik etiketlerle yeniden yaz.
2. Tıklanan şeyleri `<button>` / `<a>` yap.
3. Sonuçta görünüm aynı kaldı mı, kontrol et.

BÖLÜM 22

SEO ve Paylaşım Etiketleri

İyi HTML, sayfanın arama motorlarında bulunmasına ve sosyal medyada düzgün paylaşılmasına yardım eder. Bunun çoğu, `<head>` içindeki birkaç etiketle olur.

SEO temelleri

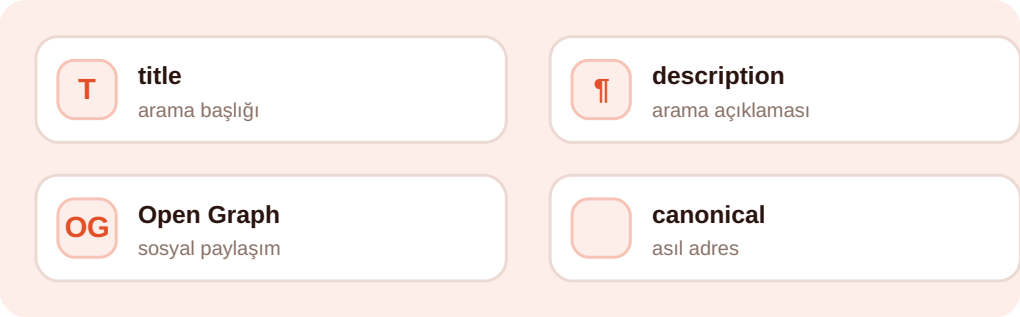
- Anlamlı bir `<title>` ve `meta description`.
- Doğru başlık yapısı (tek `<h1>`, sıralı alt başlıklar).
- Görsellerde `alt`; anlamlı bağlantı metinleri.

Sosyal paylaşım (Open Graph)

- `og:title`, `og:description`, `og:image` — paylaşımda görünen başlık, açıklama ve görsel.
- Bunlar olmadan paylaşım çirkin/eksik görünebilir.

Paylaşım etiketleri

```
<meta property="og:title" content="Taze Pastane">
<meta property="og:description" content="Her gün taze ekmek.">
<meta property="og:image" content="kapak.jpg">
```



Şema 22.1 — Arama ve paylaşım için head etiketleri.

İPUCU

SEO'nun temeli sihir değil, **kaliteli ve anlamlı HTML**'dir: doğru başlıklar, açıklamalar, alt metinleri. Open Graph etiketleri ise bağlantın paylaşıldığında profesyonel görünmesini sağlar. Bu küçük eklemeler büyük fark yaratır.

Tarayıcıda ne görünür? ÖNİZLEME

`<title>` ve `description` Google sonucunda görünür; Open Graph etiketleri ise bağlantıyı WhatsApp/sosyal medyada paylaşınca çıkan başlık, açıklama ve önizleme görselini belirler.

Alıştırma 8 dk

Paylaşıma hazırla:

1. Sayfana anlamlı `title` ve `description` ekle.
2. `og:title` ve `og:image` ekle.
3. Tek bir `<h1>` kuralına uyduğunu kontrol et.

BÖLÜM 23

Performans ve Temiz Kod

Hızlı yüklenen, temiz yazılmış sayfalar hem kullanıcıyı hem arama motorlarını memnun eder. İyi HTML alışkanlıkları, performansın temelidir.

Temiz kod alışkanlıkları

- Girintili ve düzenli yaz; her elementi doğru kapat.
- Gereksiz iç içe `<div>`'lerden kaçın.
- Anlamlı `class` adları kullan.

Görsel performansı

- Görselleri uygun boyutta kullan (devasa dosyaları küçült).
- `loading="lazy"` ile görseli görünene kadar erteleyebilirsin.
- Modern biçimler (WebP) daha küçük dosya verir.

Tembel yükleme

```
<img src="buyuk-foto.jpg" alt="Manzara"
    width="800" loading="lazy">
```

Şema 23.1 — Temiz kod ve optimize görseller hızlı sayfa demektir.

İPUCU

Performansın en büyük düşmanı genelde **kocaman görsellerdir**. Bir görseli web'e koymadan önce uygun boyuta küçült; 5 MB'lık bir fotoğraf, sayfanı saniyelerce yavaşlatır. `loading="lazy"` ise ekranda görünmeyen görselleri sonraya bırakır.

Tarayıcıda ne görünür? ÖNİZLEME

Temiz kod ve küçük görseller, sayfanın daha hızlı açılması olarak görünür (özellikle yavaş bağlantıda). `loading="lazy"` görselleri kullanıcı o noktaya kaydırınca yükler.

Alıştırma 8 dk

Hızlandır:

1. Bir HTML dosyasını girintileyip düzenle.
2. Bir görsele `loading="lazy"` ekle.
3. Gereksiz iç içe `<div>` varsa azalt.

BÖLÜM 24

Şablonlar ve Yeniden Kullanım

Profesyoneller aynı kodu defalarca yazmaz; tekrar eden parçaları şablon hâline getirir. Bu fikir, ileride bileşen (component) tabanlı geliştirmenin temelidir.

Tekrarı azaltmak

- Her sayfada aynı `<head>`, `<header>`, `<footer>` tekrarlanır.
- Bir **boilerplate** (hazır iskelet) tutup her sayfaya oradan başla.
- İleride şablon motorları/bileşenler bu tekrarı tamamen ortadan kaldırır.

Yeniden kullanılabilir parça (header)

```
<!-- her sayfada aynı header -->
<header>
  <a href="/"><img src="logo.png" alt="Logo"></a>
  <nav>...</nav>
</header>
```

Şema 24.1 — Tekrar eden parçaları şablonlaştırmak: bileşen fikrinin tohumu.

İPUCU

"Kendini tekrar etme" (DRY) ilkesi HTML'de de geçerlidir. Aynı header'ı 10 sayfaya elle kopyalarsan, bir değişiklikte 10 yeri düzeltmen gerekir. Tek bir şablon, bu derdi ortadan kaldırır — bu, bileşen tabanlı geliştirmenin de mantığıdır.

Tarayıcıda ne görünür? ÖNİZLEME

Şablon yaklaşımı görünümü değiştirmez; ama her sayfanın tutarlı görünmesini ve tek bir değişikliğin (örn. menüye yeni bağlantı) her yere kolayca yansımasını sağlar.

Alıştırma 8 dk

Şablonlaştır:

1. Sayfalarında tekrar eden bir parça (header/footer) belirle.
2. Onu tek bir "şablon parçası" olarak yaz.
3. Bunu birden çok sayfada kullanmanın faydasını yaz.

BÖLÜM 25

HTML5 API'lerine Bakış

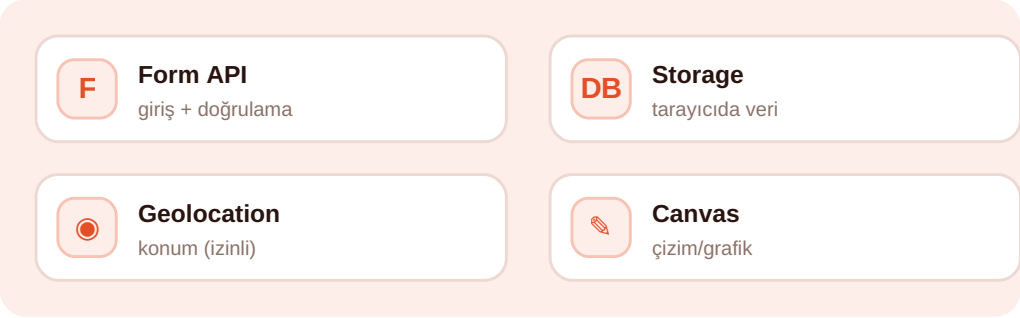
Modern HTML5, sadece etiketlerden ibaret değildir; tarayıcıya güç katan API'ler de sunar. Bunların çoğu JavaScript gerektirir; burada amacımız onları tanıyıp sonraki modüllere köprü kurmaktır.

Öne çıkan HTML5 yetenekleri

- **Form API:** gelişmiş giriş türleri ve doğrulama (gördük).
- **Depolama:** tarayıcıda küçük veri saklama (localStorage).
- **Konum (Geolocation):** kullanıcının (izinli) konumu.
- **Canvas / SVG:** çizim ve grafik.
- **Ses/Video API:** medyayı programlı kontrol.

Canvas: çizim için bir tuval (JS ile dolar)

```
<canvas id="tuval" width="300" height="150"></canvas>
<!-- içi JavaScript ile çizilir (sonraki modüller) -->
```



Şema 25.1 — HTML5'in sunduğu bazı modern yetenekler (çoğu JS ile çalışır).

İPUCU

Bu API'lerin çoğu **JavaScript ile** hayat bulur; HTML yalnızca kapıyı açar (örn. `<canvas>` boş tuvali sağlar, çizimi JS yapar). Şimdilik bunları "ileride neler mümkün?" diye tanı; JavaScript modülünde kullanmaya başlayacaksın.

Tarayıcıda ne görünür? ÖNİZLEME

`<canvas>` tek başına boş bir alan olarak görünür; içine bir şey çizilmesi için JavaScript gerekir. Bu API'ler, statik HTML'i dinamik ve güçlü uygulamalara dönüştüren köprülerdir.

Alıştırma

8 dk

Geleceğe bak:

1. Beş HTML5 yeteneğinden ikisini seç ve ne işe yaradığını yaz.
2. Bir `<canvas>` elementi ekle.
3. Bunların neden JavaScript gerektirdiğini açıkla.

BÖLÜM 26

Bitirme: Çok Sayfalı Bir Site İskeleti

Tüm HTML bilgini birleştirip birden çok sayfadan oluşan, tutarlı ve anlamlı bir site iskeleti kuruyorsun. Bu, gerçek bir web projesinin temel yapısıdır.

Çok sayfalı yapı

- `index.html` (ana sayfa), `hakkinda.html`, `iletisim.html`.
- Her sayfada tutarlı `<header>` ve `<nav>` (sayfalar arası geçiş).
- Ortak bir `styles.css` ile (sonraki modülde) tek tip görünüm.

Sayfalar arası gezinme (her sayfada nav)

```
<nav>
  <a href="index.html">Ana sayfa</a>
  <a href="hakkinda.html">Hakkında</a>
  <a href="iletisim.html">İletişim</a>
</nav>
```

Şema 26.1 — Çok sayfalı bir sitenin dosya yapısı.

İPUCU

Birden çok sayfada **aynı header/nav'ı** kullan ki kullanıcı kaybolmasın; tutarlılık iyi bir sitenin işaretidir. Sonraki modülde (CSS) bu iskelete görünüm kazandıracak, ardından (JavaScript) etkileşim ekleyeceksin. Sağlam HTML iskeleti, her şeyin temelidir.

Tarayıcıda ne görünür?

ÖNİZLEME

Çok sayfalı site, üstteki menüden (nav) sayfalar arasında gezilebilen, her sayfası tutarlı görünen bir bütün olarak görünür. Şimdilik sade; CSS ile görünüm, JS ile etkileşim eklenince tam bir siteye dönüşür.

Alıştırma

15 dk

Kendi siteni kur:

1. Üç sayfalı (ana, hakkında, iletişim) bir site iskeleti oluştur.
2. Her sayfaya aynı `<nav>`'ı ekle ve sayfaları birbirine bağla.
3. Bir sayfaya semantik bölümler ve bir form ekle.
4. Tüm sayfaları tarayıcıda gezerek test et.

EK

HTML Etiket Sözlüğü

En sık kullanılan HTML etiketleri ve işlevleri. Bir başvuru kaynağı olarak saklayabilirsin.

<!doctype html>	HTML5 belge türü
<html>	Kök element
<head> / <body>	Görünmez bilgi / görünen içerik
<h1>–<h6>	Başlık seviyeleri
<p>	Paragraf
/	Önemli / vurgu
/ /	Listeler ve öğeleri
<a href>	Bağlantı
<img src alt>	Görsel
/	Genel kapsayıcılar
Semantik etiketler	header, nav, main, footer...
<form> / <input>	Form ve giriş alanları

HTML'in özeti ÖNİZLEME

HTML, içeriği **etiketlerle** işaretleyerek sayfanın yapısını kurar: başlıklar, paragraflar, listeler, bağlantılar, görseller ve formlar. Tarayıcı bu işaretlemeyi okuyup ekrana çizer. Görünümü CSS, davranışı JavaScript ekler — ama her şey bu sağlam HTML iskeletinin üstüne kurulur.