WEB TASARIMI VE PROGRAMLAMA

LAB-2:

Semantik HTML5 · Erişilebilirlik (a11y) · Form Temelleri

İçindekiler

1 LAB-2 Öğrenme Hedefleri

İlk laboratuvarda geliştirme ortamını kurmuş, Git iş akışını öğrenmiş ve ilk React projeni oluşturmuştun. Artık araçlar hazır; şimdi sıra **web sayfalarının temelini doğru atmaya** geldi.

Bu laboratuvarda HTML'i "sadece çalışsın" yaklaşımıyla değil, **anlamlı (semantik)**, **erişilebilir (accessible)** ve **kullanıcı dostu** biçimde yazmayı öğreneceksin. Ayrıca formlarla kullanıcıdan veri toplamayı, bu verileri doğrulamayı (validation) ve hata mesajlarını yönetmeyi pratiğe dökeceksin.

Bu laboratuvarın sonunda aşağıdaki becerilere sahip olacaksın:

1. **Semantik HTML5 etiketlerini doğru kullanmak:** `<header>`, `<nav>`, `<main>`, `<section>`, `<article>`, `<footer>` gibi yapısal etiketleri yerinde ve amacına uygun kullanmak.
2. **Erişilebilirlik (a11y) ilkelerini uygulamak:** Alt metin, heading hiyerarşisi, tab ile gezinme, ARIA öznitelikleri gibi temel erişilebilirlik pratiklerini benimsemek.
3. **Form elemanlarını doğru yapılandırmak:** `<input>`, `<label>`, `<textarea>`, `<select>` gibi form bileşenlerini birbirine bağlı ve erişilebilir şekilde kullanmak.
4. **Form doğrulama ve hata mesajlarını yönetmek:** HTML5 yerleşik doğrulama özniteliklerini (`required`, `pattern`, `minlength`) kullanmak ve hata mesajı alanlarını hazırlamak.
5. **Kişisel portföy sayfası oluşturmak:** Semantik HTML ile tek sayfalık bir kişisel tanıtım sayfası ve iletişim formu geliştirmek.
6. **Lighthouse ile erişilebilirlik testi yapmak:** Chrome DevTools içindeki Lighthouse aracını çalıştırıp erişilebilirlik puanını 90+ hedefine ulaştırmak.

Benzetme ile Anlama

LAB-1'de inşaat aletlerini hazırlamıştık. Şimdi binanın temelini atıyoruz. Temeli "betona karıştırdık, dümdüz döktük" diye bırakmazsın — demirlerin doğru yerleştirilmesi, su yalıtımı, zemin etüdü gerekir. Semantik HTML ve erişilebilirlik de tam budur: Dışarıdan görünmez ama sağlamlığı ve kaliteyi belirler. Temeli yanlış atarsan, üzerine ne kadar güzel bina yaparsan yap, eninde sonunda çatlar.

Sırada Ne Var?

Hedeflerimiz belirlendi. İlk adım olarak "semantik HTML" kavramını tanıyalım: HTML etiketlerini neden rastgele değil, anlamlarına göre kullanmamız gerektiğini görelim.

2 Semantik HTML5 Nedir?

HTML etiketi yazdığında aslında tarayıcıya ve diğer yazılımlara (arama motorları, ekran okuyucular, tarayıcı eklentileri) "bu içerik ne tür bir şey" diye söylemiş olursun. Semantik HTML, bu söylemeyi **doğru ve anlamlı** yapma sanatıdır.

Teorik Arka Plan

Semantik kelimesi "anlamla ilgili" demektir. Semantik HTML, bir HTML etiketinin **içeriğin ne olduğunu** tanımlaması anlamına gelir — sadece nasıl göründüğünü değil.

Semantik olmayan (non-semantic) yaklaşım:

```
<div class="header">
  <div class="navigation">...</div>
</div>
<div class="content">...</div>
<div class="footer">...</div>
```

Yukarıdaki kodda tarayıcı, arama motoru veya ekran okuyucu "header", "navigation" veya "footer" yazdığını **anlamaz**. Bunlar sadece class isimleri — birer metin.

Semantik yaklaşım:

```
<header>
  <nav>...</nav>
</header>
<main>...</main>
<footer>...</footer>
```

Burada etiketin kendisi anlamı taşır. Tarayıcı, arama motoru ve ekran okuyucu bu alanların ne olduğunu doğrudan bilir.

2.1 Neden Semantik HTML Kullanmalıyız?

- **Erişilebilirlik:** Ekran okuyucular (screen readers) semantik etiketleri kullanarak görme engelli kullanıcılara sayfa yapısını anlatır. "Navigasyona geç", "Ana içeriğe atla" gibi kısayollar ancak semantik yapıda çalışır.
- **SEO (Arama Motoru Optimizasyonu):** Google gibi arama motorları semantik yapıyı analiz ederek sayfanın konusunu ve yapısını daha iyi anlar. Bu, arama sonuçlarında daha üst sıralara çıkmayı sağlar.
- **Bakım kolaylığı:** Kodu okuyan bir geliştirici, `<nav>` gördüğünde bunun navigasyon olduğunu anında bilir. `<div class="nav-wrapper-inner">` gördüğünde ise tahmin yürütmesi gerekir.
- **Tarayıcı özellikleri:** Bazı tarayıcılar ve eklentiler semantik etiketlere göre özel davranışlar sunar. Örneğin "okuyucu modu" (reader mode) `<article>` etiketini kullanarak ana içeriği ayırt eder.

Gerçek Hayatta Karşılığı

Bir gazete düşün. Gazetenin başlığı, köşe yazıları, haberler, reklamlar ve künye bilgisi belirli yerlerde ve belirli formatlardadır. Kimse gazetenin her yerine aynı boyutta, aynı formatta metin yazıp "class isminden anla" demez. Web sayfaları da aynı mantıkla yapılandırılmalıdır.

Sırada Ne Var?

Semantik HTML'in "ne" ve "neden" kısmını anladık. Şimdi en sık kullanılan semantik etiketlerin her birini tek tek inceleyeceğiz.

3 Semantik HTML5 Etiketleri

Aşağıda en temel semantik etiketleri, ne zaman kullanılacaklarını ve birbirleriyle ilişkilerini detaylı olarak açıklıyoruz.

Teorik Arka Plan

`<header>`
: Sayfanın veya bir bölümün başlık alanıdır. Genellikle logo, site adı, navigasyon menüsü ve arama çubuğu gibi elemanları barındırır. Bir sayfada birden fazla `<header>` olabilir (örneğin `<article>` içinde de kullanılabilir), ancak sayfanın ana başlığı genelde en dıştaki `<header>`'dır.

`<nav>`
: Navigasyon bağlantılarını gruplar. Sayfanın ana menüsü, alt menüleri veya sayfa içi bağlantıları burada yer alır. Her bağlantı grubu `<nav>` olmak zorunda değildir; yalnızca ana navigasyon blokları için kullanılmalıdır. Footer'daki birkaç bağlantı için `<nav>` kullanmak gereksizdir.

`<main>`
: Sayfanın **birincil içeriğini** saran etikettir. Bir sayfada **yalnızca bir tane** `<main>` olmalıdır. Header, footer ve sidebar gibi tekrarlayan elemanlar `<main>` dışında kalır. Ekran okuyucular "ana içeriğe atla" kısayolunu bu etikete göre belirler.

`<section>`
: Tematik olarak gruplandırılmış içerik bloğudur. Genelde bir başlık (`<h2>`, `<h3>` vb.) ile başlar. Örneğin "Hakkımda", "Projelerim", "İletişim" gibi bölümler birer `<section>` olabilir.

`<article>`
: Bağımsız, kendi başına anlam taşıyan içeriktir. Bir blog yazısı, haber makalesi, kullanıcı yorumu veya ürün kartı `<article>` ile sarılabilir. Test: "Bu içeriği sayfadan koparıp başka bir yere koyduğumda yine anlamlı olur mu?" Cevap evetse, `<article>` uygundur.

`<aside>`
: Ana içerikle dolaylı ilişkili yan içeriktir. Sidebar, ilgili yazılar kutusu veya reklam alanı gibi elemanlar burada yer alır. `<main>` içinde olabileceği gibi dışında da olabilir.

`<footer>`
: Sayfanın veya bir bölümün alt bilgi alanıdır. Telif hakkı notu, iletişim bilgileri, sosyal medya bağlantıları, site haritası bağlantıları gibi içerikler barındırır.

`<figure> ve <figcaption>`
: Görsel, diyagram, kod parçası gibi bağımsız içerikleri ve bunların açıklamalarını gruplamak için kullanılır.

Uygulama Adımı

Uygulama-1: Semantik İskelet Oluştur
Projenin `src/` klasöründe çalışarak aşağıdaki semantik yapıyı oluştur. Henüz içerik doldurmana gerek yok; **iskeleti doğru kurmaya** odaklan:

```
<header>
  <nav>
    <ul>
      <li><a href="#hakkimda">Hakkimda</a></li>
      <li><a href="#projeler">Projeler</a></li>
      <li><a href="#iletisim">Iletisim</a></li>
    </ul>
  </nav>
</header>

<main>
  <section id="hakkimda">
    <h2>Hakkimda</h2>
    <!-- icerik buraya -->
  </section>

  <section id="projeler">
    <h2>Projelerim</h2>
    <!-- icerik buraya -->
  </section>

  <section id="iletisim">
    <h2>Iletisim</h2>
    <!-- form buraya -->
  </section>
</main>

<footer>
  <p>&copy; 2025 Ad Soyad. Tum haklari saklidir.</p>
</footer>
```

Dikkat – Sık Yapılan Hatalar

- `<div>` semantik bir etiket **değildir**. Stil veya layout amacıyla kullanılır; yapısal anlam taşımaz. Mümkünse önce semantik etiketleri tercih et.
- Her `<section>` bir başlık (`<h2>`–`<h6>`) ile başlamalıdır. Başlıksız section, erişilebilirlik açısından sorunludur.
- `<main>` sayfada **sadece bir kez** kullanılmalıdır. İki tane `<main>` koymak HTML standardına aykırıdır.
- `<header>` ve `<footer>` sadece sayfa genelinde değil, `<article>` veya `<section>` içinde de kullanılabilir — ama bunu bilinçli yap.

Sırada Ne Var?

Semantik iskeleti öğrendin. Peki bu iskeleti sadece tarayıcılar değil, **engelli kullanıcılar da** doğru algılayabiliyor mu? Bir sonraki bölümde erişilebilirlik (a11y) kavramına dalacağız.

4 Erişilebilirlik (a11y) Nedir?

"a11y" kısaltması "accessibility" kelimesinin kısaltmasıdır (a + 11 harf + y). Erişilebilirlik, web içeriklerinin **engelli bireyler dahil herkes** tarafından algılanabilir, kullanılabilir ve anlaşılabilir olmasını sağlama sürecidir.

Teorik Arka Plan

Erişilebilirlik neden önemlidir?

- **Etik boyut:** Dünya nüfusunun yaklaşık %15'i bir tür engele sahiptir. Web, herkes için olmalıdır.
- **Yasal boyut:** Birçok ülkede kamu web sitelerinin erişilebilir olması yasal zorunluluktur (ABD'de ADA, AB'de EAA, Türkiye'de 5378 sayılı Kanun).
- **Teknik boyut:** Erişilebilir kod genellikle daha temiz, daha semantik ve daha bakımı kolay koddur.
- **İş boyutu:** Erişilebilir siteler daha geniş bir kitleye ulaşır ve SEO açısından avantaj sağlar.

Kimler yararlanır?

- Görme engelli kullanıcılar (ekran okuyucu kullanır)
- İşitme engelli kullanıcılar (video altyazılarına ihtiyaç duyar)
- Motor engelli kullanıcılar (fareyi kullanamaz, klavye ile gezinir)
- Bilişsel engelli kullanıcılar (basit ve tutarlı arayüz gerektirir)
- Geçici engeller (kırık kol, gürültülü ortam, parlak güneş altında ekran)
- Yaşlı kullanıcılar (azalmış görme, işitme, motor beceriler)

Benzetme ile Anlama

Bir binanın girişinde rampa olması sadece tekerlekli sandalye kullananlar için değildir: Bebek arabası iten ebeveynler, valizli yolcular, geçici sakatlığı olanlar da kullanır. Web erişilebilirliği de aynıdır — herkese fayda sağlar.

Sırada Ne Var?

Erişilebilirliğin ne olduğunu ve neden önemli olduğunu anladık. Şimdi en temel a11y tekniklerini pratikte uygulayacağız: heading hiyerarşisi, alt metin ve tab gezinme.

5 Heading Hiyerarşisi ve Alt Metin

Erişilebilirliğin en kolay ama en sık ihmal edilen iki parçası **heading hiyerarşisi** ve **alt metinlerdir**. Bunları doğru kullanmak, sayfanın erişilebilirlik puanını büyük ölçüde artırır.

5.1 Heading Hiyerarşisi

Teorik Arka Plan

HTML'de altı seviye başlık vardır: `<h1>`'den `<h6>`'ya kadar. Bu başlıklar bir **hiyerarşi** (sıradüzen) oluşturmalıdır:

- `<h1>`: Sayfanın ana başlığı. Her sayfada **yalnızca bir tane** olmalıdır.
- `<h2>`: Ana bölüm başlıkları ("Hakkımda", "Projelerim" gibi).
- `<h3>`: Alt bölüm başlıkları (`<h2>`'nin altında).
- `<h4>`–`<h6>`: Daha derin alt bölümler.

Kritik kural: Seviye atlanmamalıdır. `<h1>`'den sonra `<h3>`'e geçmek yanlıştır; önce `<h2>` kullanılmalıdır.

Derinlemesine Anlama

Ekran okuyucular, kullanıcıya sayfa yapısını heading'ler üzerinden anlatır. Kullanıcı "heading listesi"ni açarak sayfada hızlıca gezinir. Eğer heading hiyerarşisi bozuksa (mesela `<h1>` yok ama `<h3>` var), kullanıcı sayfanın yapısını anlayamaz — bu, görme engelli biri için bir kitabın bölüm başlıkları olmadan sunulması gibidir.

Doğru kullanım:

```
<h1>Ahmet Yilmaz - Kisisel Portfolyo</h1>

<h2>Hakkimda</h2>
<p>...</p>

<h2>Projelerim</h2>
  <h3>E-Ticaret Sitesi</h3>
  <p>...</p>
  <h3>Blog Uygulamasi</h3>
  <p>...</p>

<h2>Iletisim</h2>
```

Yanlış kullanım:

```
<!-- YANLIS: h1 yok, h3'ten basliyor -->
<h3>Hakkimda</h3>
<!-- YANLIS: h2'den h5'e atlandi -->
<h5>Projelerim</h5>
<!-- YANLIS: Baslik yerine kalin metin -->
<p><b>Iletisim</b></p>
```

5.2 Alt Metin (alt Attribute)

Teorik Arka Plan

Her `<img>` etiketinde `alt` özniteliği **zorunludur**. Bu öznitelik, görselin ne olduğunu metin olarak tanımlar:

- Ekran okuyucu, görseli göremez; alt metni sesli okuyarak kullanıcıya bilgi verir.
- Görsel yüklenemezse (ağ sorunu, kırık bağlantı) alt metin görüntülenir.
- Arama motorları görselleri indekslerken alt metni kullanır.

İyi alt metin kuralları:

- Görseli kısaca ve doğru tanımla: `alt="Sahilde gün batımı fotoğrafı"`
- "Resim", "Görsel", "Fotoğraf" gibi gereksiz ön ekler kullanma (ekran okuyucu zaten "image" der).
- Dekoratif görseller (arka plan deseni, ayırıcı çizgi) için boş alt kullan: `alt=""`. Bu, ekran okuyucunun görseli tamamen atlamasını sağlar.

Uygulama Adımı

Uygulama-2: Heading ve Alt Metin Kontrolü

1. Uygulama-1'de oluşturduğun iskeletin başlık hiyerarşisini kontrol et: `<h1>` ile başlıyor mu? `<h2>`'ler doğru sırada mı?
2. "Hakkımda" bölümüne bir profil fotoğrafı ekle:

```
<figure>
  <img src="profil.jpg"
       alt="Ahmet Yilmaz'in vesikalik fotografi" />
  <figcaption>Ahmet Yilmaz</figcaption>
</figure>
```

3. "Projelerim" bölümüne proje ekran görüntüleri ekle; her birine anlamlı alt metin yaz.

Dikkat – Sık Yapılan Hatalar

- `alt` özniteliğini **hiç yazmamak** ile `alt=""` yazmak farklıdır. Yazmamak hata kabul edilir; `alt=""` ise bilinçli bir tercihdir (dekoratif görsel).
- Alt metni 125 karakterin altında tut. Uzun açıklamalar için `<figcaption>` veya `aria-describedby` kullan.
- Başlıkları **asla** görsel boyut için kullanma. "Büyük görünsün" diye `<h1>` kullanmak hiyerarşiyi bozar. Boyut CSS ile ayarlanır.

Sırada Ne Var?

Heading hiyerarşisi ve alt metinlerle sayfanın yapısal erişilebilirliğini güçlendirdik. Şimdi bir adım ileri gidip **ARIA özniteliklerini** ve **klavye ile gezinme** (tab navigation) konusunu ele alacağız.

6 ARIA Öznitelikleri ve Klavye Gezinme

Semantik HTML ve doğru heading/alt metin kullanımı, erişilebilirliğin büyük kısmını çözer. Ancak bazen HTML etiketleri tek başına yeterli olmaz — özellikle özel (custom) bileşenler veya dinamik içerikler için. İşte bu noktada **ARIA** devreye girer.

6.1 ARIA Nedir?

Teorik Arka Plan

ARIA (Accessible Rich Internet Applications), HTML etiketlerine **ekstra erişilebilirlik bilgisi** ekleyen öznitelik setidir. W3C tarafından standartlaştırılmıştır.
ARIA'nın altın kuralı: Eğer bir işi native HTML ile yapabiliyorsan, ARIA kullanma. ARIA yanlış kullanıldığında erişilebilirliği iyileştirmek yerine **kötüleştirir**.
En sık kullanılan ARIA öznitelikleri:

- `aria-label`: Elemanın görsel bir etiketi olmadığında metin etiket sağlar.

```
<button aria-label="Menuyu ac">
  <span class="hamburger-icon"></span>
</button>
```

- `aria-labelledby`: Başka bir elemanın ID'sini referans alarak etiket sağlar.

```
<h2 id="projeler-baslik">Projelerim</h2>
<section aria-labelledby="projeler-baslik">...</section>
```

- `aria-describedby`: Elemana ek açıklama bağlar.

```
<input type="email" aria-describedby="email-help" />
<small id="email-help">Ornek: ad@mail.com</small>
```

- `aria-hidden="true"`: Elemanı ekran okuyucudan gizler (dekoratif ikonlar için).
- `aria-required="true"`: Bir form alanının zorunlu olduğunu bildirir (HTML5 `required` ile birlikte kullanılabilir).
- `role`: Elemanın rolünü belirtir (`role="navigation"`, `role="alert"` vb.). Semantik HTML kullanıyorsan genelde gerekli değildir.

6.2 Tab ile Gezinme (Keyboard Navigation)

Teorik Arka Plan

Motor engelli kullanıcılar fareyi kullanamaz; sayfada **Tab tuşu** ile ilerler, **Shift+Tab** ile geri gelir ve **Enter/Space** ile etkileşime girer. Bu nedenle:

- Tüm etkileşimli elemanlar (bağlantılar, butonlar, form alanları) Tab ile **erişilebilir** olmalıdır.
- Tab sırası **mantıklı** olmalıdır (sol üstten sağ alta, yukarıdan aşağıya).
- **Focus göstergesi** (genelde mavi veya siyah kenarlık) asla kaldırılmamalıdır. CSS ile `outline: none` yazmak, klavye kullanıcılarını kör eder.
- Eğer bir `<div>` veya `<span>`'ı tıklanabilir yapıyorsan, bunun yerine `<button>` veya `<a>` kullan. `<div>` Tab ile erişilemez.

Uygulama Adımı

Uygulama-3: Tab Gezinme Testi ve ARIA Ekleme

1. Sayfanı tarayıcıda aç ve fareye **hiç dokunmadan** sadece Tab tuşu ile gezinmeyi dene. Her etkileşimli elemana ulaşabiliyor musun?
2. Focus göstergesi görünüyor mu? Görünmüyorsa CSS'te `outline: none` kaldırılmış olabilir — düzelt.
3. Navigasyon menüsüne `aria-label` ekle:

```
<nav aria-label="Ana navigasyon">
  <ul>
    <li><a href="#hakkimda">Hakkimda</a></li>
    <li><a href="#projeler">Projeler</a></li>
    <li><a href="#iletisim">Iletisim</a></li>
  </ul>
</nav>
```

4. "Ana içeriğe atla" bağlantısı ekle (skip navigation):

```
<a href="#main-content" class="skip-link">
  Ana icerige atla
</a>
<!-- ... header ve nav ... -->
<main id="main-content">
  ...
</main>
```

Derinlemesine Anlama

Skip Navigation (İçeriğe Atla) Nedir?
Ekran okuyucu kullanıcıları ve klavye kullanıcıları, her sayfa yüklendiğinde navigasyon menüsünün tüm bağlantılarını Tab ile geçmek zorunda kalır. 10 menü öğesi olan bir sitede bu 10 kez Tab basmak demektir — her sayfada. "Ana içeriğe atla" bağlantısı, kullanıcının tek tuşla navigasyonu atlayıp doğrudan içeriğe geçmesini sağlar. Genelde görsel olarak gizlenir ama Tab ile focus aldığında görünür hale gelir.

Sırada Ne Var?

Erişilebilirlik araçlarını öğrendik: semantik yapı, heading hiyerarşisi, alt metin, ARIA, tab gezinme. Şimdi web'in en temel etkileşim araçlarından birine geçiyoruz: **formlar**. Kullanıcıdan veri toplamak, doğrulamak ve geri bildirim vermek formlarla olur.

7 Form Elemanları ve Label İlişkisi

Formlar, kullanıcıdan bilgi toplayan yapılardır: İletişim formu, giriş formu, arama kutusu, anket… hepsi form elemanlarından oluşur. Formları doğru ve erişilebilir kurmak, hem kullanıcı deneyimi hem de erişilebilirlik açısından kritiktir.

Teorik Arka Plan

Temel form elemanları:

`<form>`
Tüm form elemanlarını saran kapsayıcı etiket. `action` (verinin gönderileceği adres) ve `method` (`GET` veya `POST`) özniteliklerini alır.

`<input>`
En çok yönlü form elemanı. `type` özniteliği ile davranışı değişir:

- `type="text"`: Tek satır metin
- `type="email"`: E-posta (otomatik format doğrulama)
- `type="tel"`: Telefon numarası
- `type="url"`: Web adresi
- `type="password"`: Şifre (metin gizlenir)
- `type="number"`: Sayı girişi
- `type="submit"`: Gönder butonu

`<textarea>`
Çok satırlı metin alanı. Kullanıcının uzun mesaj yazması gereken yerlerde kullanılır.

`<select>` ve `<option>`
Açılır liste (dropdown). Kullanıcı önceden belirlenmiş seçeneklerden birini seçer.

`<button>`
Tıklanabilir buton. `type="submit"` (formu gönder), `type="button"` (genel buton) veya `type="reset"` (formu sıfırla) değerlerini alır.

`<fieldset>` ve `<legend>`
İlişkili form elemanlarını görsel ve semantik olarak gruplar. `<legend>` grubun başlığıdır.

7.1 Label İlişkisi: Neden Kritik?

Teorik Arka Plan

Her form alanı bir `<label>` ile **ilişkilendirilmelidir**. Bu ilişki iki şekilde kurulur:
Yöntem 1 — `for/id` ilişkisi (önerilen):

```
<label for="email-input">E-posta adresiniz:</label>
<input type="email" id="email-input" name="email" />
```

Yöntem 2 — İç içe (wrapping):

```
<label>
  E-posta adresiniz:
  <input type="email" name="email" />
</label>
```

Label ilişkisi neden zorunludur?

- **Erişilebilirlik:** Ekran okuyucu, input'a focus geldiğinde label metnini sesli okur. Label yoksa kullanıcı neyi doldurması gerektiğini bilemez.
- **Kullanılabilirlik:** Label'a tıklamak, ilişkili input'a focus verir. Bu, özellikle mobilde küçük checkbox/radio butonları için büyük kolaylıktır — tıklama alanı genişler.
- **Erişilebilirlik puanı:** Lighthouse, label'sız input'ları hata olarak raporlar.

Dikkat – Sık Yapılan Hatalar

- `placeholder` bir label **değildir**. Placeholder, kullanıcı yazmaya başlayınca kaybolur; ekran okuyucularda güvenilir şekilde okunmaz. Her zaman `<label>` kullan, placeholder'ı sadece ipucu olarak ekle.
- `<input>` elemanının `name` özniteliği form verisi gönderilirken kullanılır; `id` ise label ilişkisi ve JavaScript seçimi içindir. İkisi de gereklidir.
- `type="submit"` yerine `<button type="submit">Gönder</button>` tercih et — içerik daha esnek olur (ikon ekleyebilirsin).

Sırada Ne Var?

Form elemanlarını ve label ilişkisini öğrendik. Peki kullanıcı formu yanlış doldurduysa ne olacak? Bir sonraki bölümde **form doğrulama (validation)** ve **hata mesajı yönetimi** konusuna geçiyoruz.

8 Form Doğrulama ve Hata Mesajları

Kullanıcının forma girdiği verinin doğru formatta ve eksiksiz olduğunu kontrol etmeye **doğrulama (validation)** denir. Doğrulama iki katmanda yapılır:

Teorik Arka Plan

- **İstemci tarafı (client-side):** Veri sunucuya gönderilmeden önce tarayıcıda kontrol edilir. Hızlı geri bildirim sağlar. HTML5 öznitelikleri ve JavaScript ile yapılır.
- **Sunucu tarafı (server-side):** Veri sunucuya ulaştığında tekrar kontrol edilir. Güvenlik açısından **zorunludur** çünkü istemci tarafı doğrulama atlanabilir.

Bu LAB'da **istemci tarafı doğrulamaya** odaklanacağız.

HTML5 doğrulama öznitelikleri:

- `required`: Alan boş bırakılamaz.
- `type="email"`: E-posta formatı kontrol edilir.
- `minlength` / `maxlength`: Minimum/maksimum karakter sayısı.
- `min` / `max`: Sayısal değer aralığı.
- `pattern`: Düzenli ifade (regex) ile özel format kontrolü.

Uygulama Adımı

Uygulama-4: Doğrulamalı İletişim Formu
Aşağıdaki formu "İletişim" bölümüne yerleştir. Her alana uygun doğrulama özniteliklerini ve hata mesajı alanlarını ekle:

```
<form action="#" method="POST" novalidate>
  <fieldset>
    <legend>Iletisim Formu</legend>

    <div class="form-group">
      <label for="name">Ad Soyad:</label>
      <input type="text" id="name" name="name"
             required minlength="2"
             aria-describedby="name-error" />
      <small id="name-error" class="error-msg"
             role="alert"></small>
    </div>

    <div class="form-group">
      <label for="email">E-posta:</label>
      <input type="email" id="email" name="email"
             required
             aria-describedby="email-error" />
      <small id="email-error" class="error-msg"
             role="alert"></small>
    </div>

    <div class="form-group">
      <label for="subject">Konu:</label>
      <select id="subject" name="subject" required
              aria-describedby="subject-error">
        <option value="">-- Seciniz --</option>
        <option value="is">Is Teklifi</option>
        <option value="soru">Soru</option>
        <option value="oneri">Oneri</option>
      </select>
      <small id="subject-error" class="error-msg"
             role="alert"></small>
    </div>

    <div class="form-group">
      <label for="message">Mesajiniz:</label>
      <textarea id="message" name="message"
                rows="5" required minlength="10"
                aria-describedby="message-error">
      </textarea>
      <small id="message-error" class="error-msg"
             role="alert"></small>
    </div>

    <button type="submit">Gonder</button>
  </fieldset>
</form>
```

Derinlemesine Anlama

`novalidate` ve `role="alert"` Neden Var?
`novalidate` özniteliği tarayıcının varsayılan doğrulama balonlarını devre dışı bırakır. Böylece kendi özel hata mesajlarımızı gösterebiliriz (JavaScript ile — ilerleyen LAB'larda yapılacak). Şimdilik hata mesajı alanlarını **hazır tutuyoruz**.
`role="alert"`, içerik değiştiğinde ekran okuyucunun bunu anında sesli olarak okumasını sağlar. Kullanıcı formu göndermeye çalıştığında hata mesajı eklendiğinde, ekran okuyucu "Ad Soyad alanı zorunludur" gibi bir metni otomatik olarak okur.

Dikkat – Sık Yapılan Hatalar

- `novalidate` kullandıysan, JavaScript ile kendi doğrulama mantığını yazmayı **unutma**. Aksi takdirde hiçbir doğrulama çalışmaz. Bu LAB'da HTML yapısını hazırlıyoruz; JavaScript doğrulaması sonraki LAB'da eklenecek.
- Hata mesajlarını `placeholder` içine yazma. Placeholder kullanıcı yazmaya başlayınca kaybolur; hata mesajı her zaman görünür olmalıdır.
- Form alanlarının `name` özniteliklerini anlamlı tut: `name="user_email"` iyidir, `name="field3"` kötüdür.

Sırada Ne Var?

Form yapısını, doğrulama özniteliklerini ve hata mesajı alanlarını öğrendik. Şimdi tüm bu bilgileri bir araya getirip **kişisel portföy sayfasını** tam olarak inşa edeceğiz.

9 Uygulama: Kişisel Portföy Sayfası

Bu bölümde, öğrendiğin tüm kavramları birleştirerek tek sayfalık bir kişisel portföy web sayfası oluşturacaksın. Sayfa semantik HTML ile yapılandırılacak, erişilebilirlik ilkelerine uygun olacak ve bir iletişim formu içerecek.

Uygulama Adımı

Uygulama-5: Portföy Sayfasını Tamamla
Aşağıdaki adımları sırayla uygula:

1. **Header ve Navigasyon:**
   - Adını veya logonu içeren bir `<header>` oluştur.
   - Sayfa içi bağlantılar içeren bir `<nav>` ekle (`aria-label` ile etiketle).
   - "Ana içeriğe atla" bağlantısını (skip link) ekle.
2. **Hakkımda Bölümü** (`<section id="hakkimda">`):
   - `<h2>` ile bölüm başlığı.
   - Profil fotoğrafı (`<figure>` + `<img alt="...">` + `<figcaption>`).
   - Kısa bir tanıtım paragrafı.
   - Kullandığın teknolojiler listesi (`<ul>`).
3. **Projelerim Bölümü** (`<section id="projeler">`):
   - En az 2 proje kartı. Her biri bir `<article>` içinde:
   - `<h3>` ile proje adı, kısa açıklama, kullanılan teknolojiler.
   - Varsa ekran görüntüsü (alt metinli).
4. **İletişim Bölümü** (`<section id="iletisim">`):
   - Uygulama-4'te oluşturduğun doğrulamalı iletişim formunu buraya yerleştir.
5. **Footer:**
   - Telif hakkı bilgisi.
   - Sosyal medya bağlantıları (varsa).

Uygulama Adımı

Uygulama-6: Temel CSS ile Stil Ver
Portföy sayfasını sadece HTML ile bırakmak yeterli değil; okunabilirlik ve kullanılabilirlik için minimal CSS ekle:

```css
/* Genel sifirlama */
* { margin: 0; padding: 0; box-sizing: border-box; }
body { font-family: system-ui, sans-serif; line-height: 1.6; }

/* Skip link: normalde gizli, Tab ile gorunur */
.skip-link {
  position: absolute;
  top: -100px;
  left: 0;
  background: #1E3A8A;
  color: white;
  padding: 8px 16px;
  z-index: 100;
}
.skip-link:focus { top: 0; }

/* Focus gostergesi -- ASLA kaldirma */
:focus { outline: 2px solid #2563EB; outline-offset: 2px; }

/* Form stilleri */
.form-group { margin-bottom: 1rem; }
label { display: block; margin-bottom: 0.25rem; }
input, textarea, select {
  width: 100%;
  padding: 0.5rem;
  border: 1px solid #ccc;
  border-radius: 4px;
}
.error-msg { color: red; font-size: 0.85rem; }
```

Sırada Ne Var?

Portföy sayfası hazır! Ama erişilebilirlik iddiamızı kanıtlamamız gerekiyor. Bir sonraki bölümde Chrome DevTools'un **Lighthouse** aracını kullanarak erişilebilirlik puanını ölçeceğiz.

10 Lighthouse ile Erişilebilirlik Testi

Erişilebilirlik kurallarını uyguladığını **ölçülebilir** şekilde doğrulamak için Google'ın **Lighthouse** aracını kullanacağız. Lighthouse, Chrome tarayıcısına entegre bir denetim aracıdır.

Teorik Arka Plan

Lighthouse aşağıdaki kategorilerde sayfa analizi yapar:

- **Performance:** Sayfa yüklenme hızı.
- **Accessibility:** Erişilebilirlik uyumu.
- **Best Practices:** Genel iyi pratikler.
- **SEO:** Arama motoru optimizasyonu.

Bu LAB'da odağımız **Accessibility (Erişilebilirlik)** kategorisidir. Hedefimiz **90+** puan almak.

Uygulama Adımı

Uygulama-7: Lighthouse Raporu Oluştur

1. Projeyi `npm run dev` ile çalıştır ve sayfayı **Google Chrome**'da aç.
2. Sağ tıkla → **Inspect** (veya `F12`) ile DevTools'u aç.
3. Üst menüde **Lighthouse** sekmesine tıkla (görünmüyorsa `>>` simgesine tıklayarak bul).
4. Ayarlar:
   - **Mode:** Navigation
   - **Device:** Desktop (veya Mobile)
   - **Categories:** Sadece **Accessibility** işaretle (diğerlerini kaldır, böylece daha hızlı çalışır).
5. **Analyze page load** butonuna tıkla.
6. Rapor oluşturulunca erişilebilirlik puanını not et.
7. Eğer puan 90'ın altındaysa, rapordaki hata ve uyarıları oku; her birini düzelt ve tekrar çalıştır.

Derinlemesine Anlama

Lighthouse Hangi Kontrolleri Yapar?

Erişilebilirlik kategorisinde Lighthouse şunları kontrol eder (tam olmayan liste):

- Görsellerde alt metin var mı?
- Form alanlarında label var mı?
- Heading hiyerarşisi mantıklı mı (seviye atlama yok mu)?
- Metin/arka plan renk kontrastı yeterli mi (en az 4.5:1 oranı)?
- Sayfa dili belirtilmiş mi (`<html lang="tr">`)?
- Bağlantı metinleri anlamlı mı ("tıklayın" yerine "portföyü görüntüle" gibi)?
- ARIA öznitelikleri doğru kullanılmış mı?
- Tab sırası mantıklı mı?

Dikkat – Sık Yapılan Hatalar

- Lighthouse 100 puan vermesi, sitenin %100 erişilebilir olduğu anlamına **gelmez**. Lighthouse otomatik olarak kontrol edilebilecek kuralları test eder; manuel test (gerçek ekran okuyucu ile deneme) de gereklidir.
- Lighthouse'u **Incognito (Gizli) modda** çalıştırmak daha güvenilir sonuçlar verir çünkü tarayıcı eklentileri sonuçları etkileyebilir.
- Renk kontrastı hatası alıyorsan, arka plan ile metin rengi arasındaki kontrastı artır. https://webaim.org/resources/contrastchecker/ adresinden kontrol edebilirsin.

Sırada Ne Var?

Lighthouse raporu aldık ve erişilebilirlik puanımızı gördük. Şimdi tüm çalışmayı Git ile kayıt altına alıp GitHub'a yükleyelim ve teslim kontrol listesine geçelim.

11 Git İş Akışı ve Teslim

LAB-1'de öğrendiğin Git iş akışını bu LAB'da da uygulayacaksın. Kısa bir hatırlatma:

Uygulama Adımı

Uygulama-8: Git ile Kaydet ve GitHub'a Yükle

1. Proje klasöründe (veya LAB-1'deki mevcut repoda) yeni bir branch aç:

```
git checkout -b feature/semantic-portfolio
```

2. Değişiklikleri commit et (birden fazla küçük commit tercih edilir):

```
git add .
git commit -m "feat: add semantic HTML portfolio structure"
```

3. Stil ve form ekledikçe ayrı commitler at:

```
git add .
git commit -m "feat: add accessible contact form"

git add .
git commit -m "style: add base CSS and skip link"

git add .
git commit -m "fix: improve a11y based on Lighthouse report"
```

4. GitHub'a push et:

```
git push -u origin feature/semantic-portfolio
```

5. Lighthouse raporunun ekran görüntüsünü alıp repoya ekle veya README'ye yapıştır.

12 Teslim Kontrol Listesi ve Değerlendirme Rubriği

Kontrol Listesi

Teslim öncesi kontrol listesi:

- ☐ Sayfa semantik etiketlerle yapılandırılmış mı? (`header`, `nav`, `main`, `section`, `article`, `footer`)
- ☐ Heading hiyerarşisi doğru mu? (`h1` → `h2` → `h3`, seviye atlanmamış)
- ☐ Tüm görsellerde anlamlı `alt` metni var mı?
- ☐ İletişim formu mevcut ve tüm input'larda `<label>` ilişkisi kurulmuş mu?
- ☐ Form alanlarında `required`, `minlength`, `type` gibi doğrulama öznitelikleri var mı?
- ☐ Hata mesajı alanları (`<small>` + `role="alert"`) hazırlanmış mı?
- ☐ Tab ile gezinme sorunsuz çalışıyor mu? Focus göstergesi görünüyor mu?
- ☐ Skip navigation bağlantısı eklenmiş mi?
- ☐ `<html lang="tr">` ayarlanmış mı?
- ☐ Lighthouse erişilebilirlik puanı **90+** mı? (Ekran görüntüsü eklenmeli)
- ☐ En az 3 commit atılmış ve branch kullanılmış mı?
- ☐ GitHub'a push edilmiş mi?

12.1 Değerlendirme Rubriği

Kriter	Puan	Açıklama
Semantik HTML yapısı doğru	20	header/nav/main/section/footer
Heading hiyerarşisi ve alt metinler	10	h1–h3 sıralı, alt anlamlı
Form yapısı ve label ilişkisi	15	label + for/id eşleşmesi
Form doğrulama öznitelikleri	10	required, minlength, type
Hata mesajı alanları hazır	5	role=”alert”, aria-describedby
ARIA kullanımı ve skip link	10	aria-label, skip navigation
Tab gezinme ve focus göstergesi	10	Klavye ile tam erişim
Lighthouse erişilebilirlik puanı 90+	10	Ekran görüntüsü ile kanıt
Git iş akışı (branch + commit)	10	En az 3 anlamlı commit
Toplam	**100**	

13 Kendini Test Et

Kendini Test Et

Aşağıdaki sorulara kendi cümlelerinle 2–3 satır cevap yazabiliyor musun? Yazamıyorsan ilgili bölümü tekrar oku.

1. **Semantik HTML ne demektir? `<div>` ile `<section>` arasındaki fark nedir?**
2. **`<main>` etiketi neden sayfada yalnızca bir kez kullanılmalıdır?**
3. **Erişilebilirlik (a11y) nedir ve kimler yararlanır?**
4. **Heading hiyerarşisinde seviye atlamak neden sorunludur?**
5. **`alt=""` ile `alt` özniteliğini hiç yazmamak arasındaki fark nedir?**
6. **ARIA ne zaman kullanılmalıdır? "Altın kuralı" nedir?**
7. **`<label>` ile form input ilişkisi neden zorunludur? `placeholder` neden yeterli değildir?**
8. **`required`, `minlength` ve `pattern` öznitelikleri ne işe yarar?**
9. **Tab ile gezinme neden önemlidir? Focus göstergesi neden kaldırılmamalıdır?**
10. **Lighthouse erişilebilirlik puanı 100 olması, sitenin tamamen erişilebilir olduğu anlamına gelir mi?**

14 Ek Kaynaklar ve İleri Okuma

- **HTML Semantik Referans:** https://developer.mozilla.org/en-US/docs/Glossary/Semantics — MDN'de semantik HTML rehberi.
- **WCAG 2.1 Rehberi:** https://www.w3.org/WAI/WCAG21/quickref/ — Web erişilebilirlik standartları.
- **a11y Proje:** https://www.a11yproject.com — Pratik erişilebilirlik ipuçları ve kontrol listesi.
- **ARIA Başlangıç:** https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA — ARIA öznitelikleri referansı.
- **Renk Kontrast Kontrol:** https://webaim.org/resources/contrastchecker/ — Metin/arka plan kontrast oranı hesaplayıcı.
- **Ekran Okuyucu Deneyimi:** https://www.nvaccess.org — NVDA (ücretsiz ekran okuyucu) ile kendi sayfanı test et.
- **HTML Form Referans:** https://developer.mozilla.org/en-US/docs/Learn/Forms — MDN form rehberi.