

DOM Manipülasyonu ve Olay Yönetimi

- /
- Mobil menü
- Element oluşturma

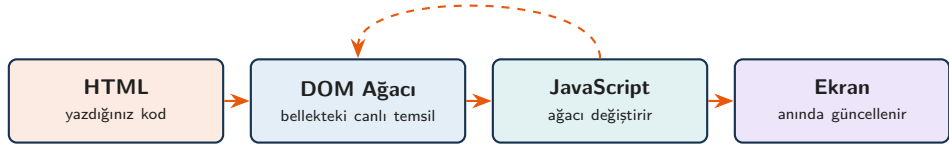
Prof. Dr. Özal YILDIRIM

Fırat Üniversitesi — Teknoloji Fakültesi

Yazılım Mühendisliği Bölümü

Sonunda: JavaScript Sayfayı Değiştiriyor

Hafta 10–11’de öğrendiklerimiz hep **konsolda** kaldı. Bu hafta o kod **ekrana çıkıyor**.



İpucu

Kritik kavrayış: JavaScript, HTML **dosyasını** değiştirmez — **bellekteki DOM ağacını** değiştirir. Sayfayı yenilerseniz tüm değişiklikler **kaybolur**.

- ▶ `querySelector` ve `querySelectorAll` ile elementlere erişebilecek,
- ▶ `textContent`, `innerHTML` ve `value` farkını açıklayabilecek,
- ▶ `classList` ile CSS sınıflarını yönetebilecek,
- ▶ `createElement` ile dinamik olarak element oluşturabilecek,
- ▶ `addEventListener` ile olayları dinleyebilecek,
- ▶ Olay nesnesini (`event`) ve `preventDefault()` kullanımını kavrayabilecek,
- ▶ Olay delegasyonu (event delegation) tekniğini uygulayabilecek,
- ▶ BookVerse mobil menüsünü ve dinamik kitap listesini hayata geçirebileceksiniz.

Bu Haftanın İçeriği

- 1 Elementlere Erişim
- 2 Elementleri Değiştirmek
- 3 Olaylar
- 4 Uygulama: BookVerse
- 5 Yapay Zeka ile DOM
- 6 Özet ve Değerlendirme



Elementlere Erişim

BÖLÜM 1

document Nesnesi ve Seçiciler

```
1 // Eslesen ilk elementi dondurur (yoksa null)
2 const anaBaslik = document.querySelector("h1");
3 const navMenu   = document.querySelector("header nav");
4 const buton     = document.querySelector("#menu-button");
5 const ilkKart   = document.querySelector(".kitap-kart");
6
7 // Eslesen TUM elementleri dondurur (NodeList - bos olabilir)
8 const menuElemanlari = document.querySelectorAll("nav li");
9 const tumKartlar      = document.querySelectorAll(".kitap-kart");
10
11 console.log(tumKartlar.length);           // kac tane bulundu
12
13 // NodeList üzerinde gezinme
14 tumKartlar.forEach(kart => console.log(kart.textContent));
15
16 // Bir elementin ICINDE arama
17 const ilkBaslik = ilkKart.querySelector("h3");
```

CSS'te öğrendiğiniz tüm seçiciler burada da geçerlidir: `"nav > ul li:first-child", "[data-id = "3"]", ".kart.aktif"...`

Eski ve Yeni Seçim Yöntemleri

Metot	Döndürdüğü	Not
<code>querySelector(sec)</code>	Tek element / <code>null</code>	Tercih edin
<code>querySelectorAll(sec)</code>	<code>NodeList</code> (statik)	Tercih edin , <code>forEach</code> çalışır
<code>getElementById(id)</code>	Tek element	En hızlısı, <code>#</code> yazılmaz
<code>getElementsByClassName</code>	<code>HTMLCollection</code> (canlı)	<code>forEach</code> çalışmaz
<code>getElementsByTagName</code>	<code>HTMLCollection</code> (canlı)	Eski kod

⚠ En Sık Hata

`querySelector` bulamazsa `null` döner.
`null.textContent` yazarsanız: `TypeError: Cannot read properties of null` **Sebebi**
genelde: yazım hatası, `#` unutmak veya `script`'in

🛡 Güvenli Yazım

```
const buton = document.querySelector("#menu");

if (buton) {
  buton.addEventListener("click", ac);
}
```

Prof. Dr. Özal YILDIRIM



Elementleri Değiştirmek

BÖLÜM 2

```
1  const baslik = document.querySelector("h1");
2
3  // 1) textContent - SADECE METİN (guvenli, hizli)
4  baslik.textContent = "BookVerse'e Hos Geldiniz!";
5  console.log(baslik.textContent);
6
7  // 2) innerHTML - HTML olarak yorumlanır (DIKKATLİ KULLANIN)
8  baslik.innerHTML = "BookVerse <span class='vurgu'>Yeni!</span>";
9
10 // 3) value - SADECE form elemanlari icin
11 const eposta = document.querySelector("#emailInput");
12 console.log(eposta.value);           // kullanicinin yazdigi
13 eposta.value = "ornek@mail.com";    // deger atama
14
15 // 4) Oznitelik islemleri
16 const gorsel = document.querySelector("img");
17 gorsel.src = "images/yeni.jpg";
18 gorsel.alt = "Yeni kapak";
19 gorsel.setAttribute("loading", "lazy");
20 gorsel.getAttribute("src");
21 gorsel.dataset.kitapId = "3";       // <img data-kitap-id="3">
```

✗ Doğrudan stil (kaçının)

```
kart.style.backgroundColor = "#f8f9fa";  
kart.style.display = "none";  
kart.style.borderRadius = "8px";  
// CSS: background-color -> JS: backgroundColor
```

Stil bilgisi **JS'e dağılır**, bakımı zorlaştır.

✓ Sınıf yönetimi (tercih edin)

```
kart.classList.add("aktif");  
kart.classList.remove("gizli");  
kart.classList.toggle("acik"); // varsa sil, yoksa  
// ekle  
kart.classList.contains("aktif"); // true/false  
kart.classList.replace("eski", "yeni");
```

Stil **CSS'te kalır**, JS sadece **durum** yönetir.

```
/* Stil CSS dosyasında tanımlı */  
.kitap-kart { transition: box-shadow 0.2s; }  
.kitap-kart.aktif { box-shadow: 0 4px 16px rgb(0 0 0 / 0.15); }  
.kitap-kart.gizli { display: none; }
```

Altın kural: JavaScript **durumu** değiştirsin (`classList.toggle`), **görünümü** CSS belirlesin.

Element Oluşturma ve Ekleme

```
1 // 1) Oluştur
2 const kart = document.createElement("article");
3
4 // 2) Yapılandır
5 kart.className = "kitap-kart";
6 kart.dataset.id = 3;
7 kart.innerHTML = `
8   <h3>${kitap.ad}</h3>
9   <p>${kitap.yazar}</p>
10  <span class="fiyat">${kitap.fiyat.toFixed(2)} TL</span>
11 `;
12
13 // 3) Sayfaya ekle
14 const liste = document.querySelector(".kitap-listesi");
15 liste.append(kart);           // SONA ekle
16 liste.prepend(kart);         // BASA ekle
17 liste.before(kart);          // ONCESINE
18 liste.after(kart);           // SONRASINA
19
20 // Silme ve temizleme
21 kart.remove();               // kendini sil
22 liste.replaceChildren();      // tum cocuklari temizle
```

Performans: DocumentFragment

✗ Yavaş — 100 kez reflow

```
kitaplar.forEach(kitap => {  
  const kart = kartOlustur(kitap);  
  liste.append(kart);      // her seferinde  
});                        // DOM guncellenir
```

✓ Hızlı — tek reflow

```
const parca = document.createDocumentFragment();  
  
kitaplar.forEach(kitap => {  
  parca.append(kartOlustur(kitap));  
});  
  
liste.append(parca);      // TEK seferde
```

```
1 // Alternatif: once metni olustur, sonra tek seferde bas  
2 function listeyiCiz(liste, kitaplar) {  
3   liste.innerHTML = kitaplar  
4     .map(k => `  
5       <article class="kitap-kart" data-id="${k.id}">  
6         <h3>${k.ad}</h3>  
7         <p>${k.yazar}</p>  
8         <span class="fiyat">${k.fiyat.toFixed(2)} TL</span>  
9       </article>`)  
10    .join("");  
11 }
```

Hafta 1'de öğrendiğimiz **Layout (reflow)** adımı pahalı bir işlemdir — her ekleme onu yeniden tetikler.



Olaylar

BÖLÜM 3

addEventListener()

```
1  const benimButonum = document.querySelector("#buton");
2
3  function butonTiklandi() {
4      console.log("Butona tiklandi!");
5  }
6
7  benimButonum.addEventListener("click", butonTiklandi);
8
9  // Ok fonksiyonu ile (daha yaygın)
10 benimButonum.addEventListener("click", () => {
11     console.log("Butona tiklandi!");
12 });
13
14 // Dinleyiciyi kaldırma (aynı fonksiyon referansı gerekir)
15 benimButonum.removeEventListener("click", butonTiklandi);
16
17 // Sadece BİR KEZ çalışsın
18 benimButonum.addEventListener("click", butonTiklandi, { once: true });
```

İki argüman: (1) olayın türü (metin), (2) olay gerçekleştiğinde çalışacak **geri çağırma fonksiyonu (callback)**.

Yaygın Olay Türleri

Olay	Ne zaman?	Olay	Ne zaman?
click	Tıklama	submit	Form gönderimi
dblclick	Çift tıklama	input	Yazılırken (her tuş)
mouseover	Fare üzerine gelince	change	Değer değişip odak çıkınca
mouseout	Fare ayrılınca	focus	Alana odaklanınca
keydown	Tuşa basılınca	blur	Odak kaybolunca
keyup	Tuş bırakılınca	scroll	Sayfa kaydırılınca
load	Tüm içerik yüklenince	resize	Pencere boyutu değişince
DOMContentLoaded	DOM hazır olunca	copy	Kopyalama yapılınca

```
// DOM hazır olunca çalıştır (script <head>'de ise sart)
```

```
document.addEventListener("DOMContentLoaded", () => {
```

Olay Nesnesi (event)

```
1 buton.addEventListener("click", (event) => {
2   console.log(event.type);           // "click"
3   console.log(event.target);         // olayin GERCEKTE olustugu element
4   console.log(event.currentTarget); // dinleyicinin BAGLI oldugu element
5   console.log(event.target.dataset.id);
6 });
7
8 // Form gonderimini engelleme - COK ONEMLI
9 form.addEventListener("submit", (event) => {
10  event.preventDefault();           // sayfanin yenilenmesini ENGELLE
11  const eposta = form.querySelector("#emailInput").value;
12  if (!eposta.includes("@")) {
13    hataGoster("Gecerli bir e-posta giriniz");
14    return;
15  }
16  basariGoster("Kaydiniz alindi!");
17 });
18
19 // Klavye olaylari
20 input.addEventListener("keydown", (e) => {
21  if (e.key === "Enter") ara();
22  if (e.key === "Escape") temizle();
23 });
```

Olay Delegasyonu (Event Delegation)

100 kitap kartına **100 ayrı dinleyici** eklemek yerine, **tek bir dinleyici** ile hepsini yönetin.

✗ Verimsiz

```
document.querySelectorAll(".kitap-kart")
  .forEach(kart => {
    kart.addEventListener("click", () => {
      console.log(kart.dataset.id);
    });
  });
// Sonradan EKLENEN kartlar çalışmaz!
```

✓ Delegasyon

```
const liste = document.querySelector(".kitap-listesi");

liste.addEventListener("click", (e) => {
  const kart = e.target.closest(".kitap-kart");
  if (!kart) return;
  console.log(kart.dataset.id);
});
// Sonradan eklenenler DE çalışır
```

💡 İpucu

Nasıl çalışır? Olaylar DOM ağacında **yukarı doğru kabarcıklanır** (*bubbling*). Kartın içindeki başlığa tıklarsanız bile olay **listeye kadar** çıkar. `e.target.closest(".kitap-kart")` ile “tıklanan öğenin en yakın kart atasını” buluruz.



Uygulama: BookVerse

BÖLÜM 4

Mobil Menü — 1. Adım: HTML

```
1 <header>
2   
3
4   <button id="menu-button"
5         aria-expanded="false"
6         aria-controls="main-nav">
7     Menu
8   </button>
9
10  <nav id="main-nav" aria-label="Ana menu">
11    <ul role="list">
12      <li><a href="index.html">Ana Sayfa</a></li>
13      <li><a href="kitaplar.html">Kitaplar</a></li>
14      <li><a href="hakkimizda.html">Hakkimizda</a></li>
15      <li><a href="iletisim.html">Iletisim</a></li>
16    </ul>
17  </nav>
18 </header>
```

id, elemente verilen **benzersiz kimliktir** — JavaScript ile kolayca seçmek için ekledik. **aria-expanded** ise ekran okuyucuya menünün **açık mı kapalı mı** olduğunu bildirir.

Mobil Menü — 2. Adım: CSS

```
1 #menu-button {
2     display: none;                /* Genis ekranda gizli */
3     background: #0056b3;
4     color: white;
5     border: none;
6     padding: 10px 15px;
7     cursor: pointer;             /* Uzerine gelince el isareti */
8     border-radius: 6px;
9 }
10
11 /* --- Mobil cihazlar icin --- */
12 @media (max-width: 768px) {
13     header { flex-direction: column; align-items: flex-start; }
14     #menu-button { display: block; } /* Butonu goster */
15     #main-nav { display: none; width: 100%; } /* Menuyu gizle */
16     #main-nav.acik { display: block; } /* .acik sinifi varsa goster */
17     #main-nav ul { flex-direction: column; gap: 0; }
18     #main-nav a { display: block; padding: 12px 0; }
19 }
```

Kritik satır: `#main-nav.acik { display: block; }` — JavaScript sadece `.acik` sınıfını ekleyip kaldıracak.

Mobil Menü — 3. Adım: JavaScript

```
1  const menuButonu = document.querySelector("#menu-button");
2  const anaMenu    = document.querySelector("#main-nav");
3
4  if (menuButonu && anaMenu) {
5    menuButonu.addEventListener("click", () => {
6      // 1) Sinifi ac/kapa
7      const acikMi = anaMenu.classList.toggle("acik");
8
9      // 2) Erisilebilirlik durumunu guncelle
10     menuButonu.setAttribute("aria-expanded", acikMi);
11
12     // 3) Buton metnini degistir
13     menuButonu.textContent = acikMi ? "Kapat" : "Menu";
14   });
15 }
```

💡 İpucu

`classList.toggle()` **true/false döndürür**
— sınıf eklendiyse **true**. Bunu doğrudan **aria-expanded** değerine verebiliriz.

✅ Toplam Sonuç

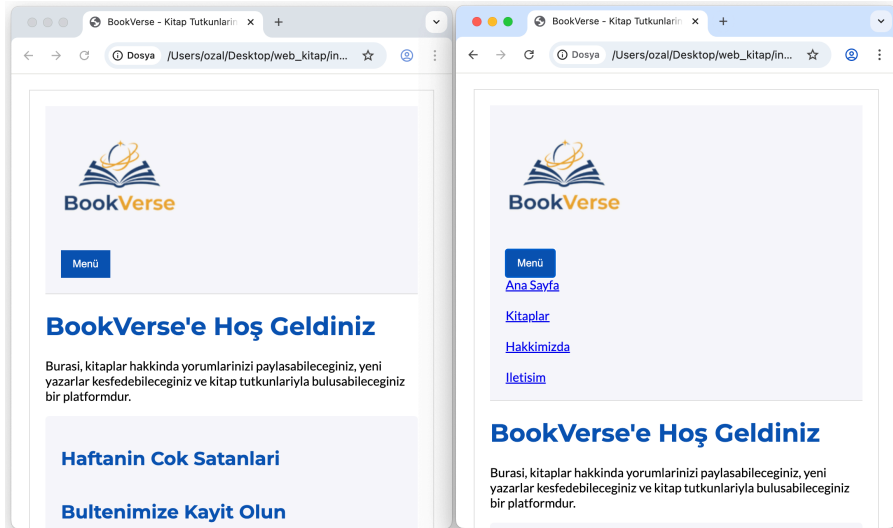
9 satır JavaScript ile projemizin ilk **interaktif özelliği** tamamlandı. Stil CSS'te, davranış JS'te, yapı HTML'de — **temiz avrım**

Dinamik Kitap Listesi

```
1 // Hafta 11'deki katalog.js verisini kullanıyoruz
2 const liste = document.querySelector(".kitap-listesi");
3
4 function kartHTML(k) {
5   return `
6     <article class="kitap-kart" data-id="${k.id}">
7       <h3>${k.ad}</h3>
8       <p class="yazar">${k.yazar}</p>
9       <p class="fiyat">${k.fiyat.toFixed(2)} TL</p>
10      <span class="puan">${k.puan}</span>
11      ${k.stok === 0 ? '<span class="etiket">Stokta yok</span>' : ""}
12    </article>`;
13 }
14
15 function listeyiCiz(veri) {
16   if (veri.length === 0) {
17     liste.innerHTML = "<p class='bos'>Aradiginiz kriterlere uygun kitap bulunamadi.</p>";
18     return;
19   }
20   liste.innerHTML = veri.map(kartHTML).join("");
21 }
22
23 // İlk çizim
24 listeyiCiz(kitaplar);
25
26 // Canlı arama
27 const aramaKutusu = document.querySelector("#arama");
28 aramaKutusu.addEventListener("input", (e) => {
29   const terim = e.target.value.toLocaleLowerCase("tr");
30   listeyiCiz(kitaplar.filter(k =>
31     k.ad.toLocaleLowerCase("tr").includes(terim) ||
32     k.yazar.toLocaleLowerCase("tr").includes(terim) ||
33     k.fiyat.toLocaleLowerCase("tr").includes(terim) ||
34     k.puan.toLocaleLowerCase("tr").includes(terim) ||
35     k.stok.toLocaleLowerCase("tr").includes(terim)
36   ));
37 });
```

Form Doğrulama

```
1  const form    = document.querySelector("#bulten-formu");
2  const eposta  = document.querySelector("#emailInput");
3  const uyari   = document.querySelector("#form-mesaj");
4
5  form.addEventListener("submit", (e) => {
6    e.preventDefault();                // sayfa YENILENMESIN
7
8    const deger = eposta.value.trim();
9
10   if (deger === "") {
11     mesajGoster("E-posta adresi bos birakilamaz.", "hata");
12     eposta.focus();
13     return;
14   }
15   if (!deger.includes("@") || !deger.includes(".")) {
16     mesajGoster("Gecerli bir e-posta adresi giriniz.", "hata");
17     return;
18   }
19
20   mesajGoster("Kaydiniz alindi. Tesekkurler!", "basari");
21   form.reset();
22 });
```



DOM'da Gezinme (Traversal)

```
1  const kart = document.querySelector(".kitap-kart");
2
3  // YUKARI - atalar
4  kart.parentElement;           // dogrudan ust element
5  kart.closest(".kitap-listesi"); // en yakin eslesen ata (KENDISI dahil)
6
7  // ASAGI - cocuklar
8  kart.children;                // tum cocuk ELEMENTLER (HTMLCollection)
9  kart.firstElementChild;
10 kart.lastElementChild;
11 kart.querySelector("h3");     // icinde arama
12
13 // YANLAR - kardesler
14 kart.nextElementSibling;      // sonraki kardes
15 kart.previousElementSibling;  // onceki kardes
16
17 // Kontrol
18 kart.matches(".aktif");       // bu secici ile eslesiyor mu?
19 kart.contains(digerElement);  // icinde barindiriyor mu?
```

HTML elementlerine **özel veri** ilâştirmenin standart yolu.

```
<article class="kitap-kart"
  data-id="3"
  data-tur="roman"
  data-stok="12">
  <h3>Tutunamayanlar</h3>
  <button class="sepete-ekle" data-fiyat="120.00">Sepete Ekle</button>
</article>
```

```
const kart = document.querySelector(".kitap-kart");

kart.dataset.id;      // "3"      (HER ZAMAN metin doner!)
kart.dataset.tur;     // "roman"
Number(kart.dataset.stok); // 12

kart.dataset.stok = 11;      // guncelleme
delete kart.dataset.tur;    // silme

// data-kitap-id -> dataset.kitapId (kebab-case -> camelCase)
```

Neden kullanılır? DOM elementi ile JavaScript verisi arasında **köprü** kurar. Tıklanan kartın *hangi kitap*

>_ Konsol Numaraları

```
$0           // secili element (Elements'te)
$("#h1")     // querySelector kisayolu
$$(".kart")   // querySelectorAll kisayolu

// Bir elementin dinleyicilerini gor
getEventListeners($0)

// Elemente kirlma noktasi koy
// (Elements > sag tik > Break on)
```

🔧 Sources Panelinde Debug

- 1 Sources sekmesi → `main.js`
- 2 Satır numarasına tıklayıp **kırılma noktası** koyun
- 3 Sayfayı etkileşime sokun — kod durur
- 4 Sağ panelde **değişken değerlerini** görün
- 5 F10 ile satır satır ilerleyin

💡 İpucu

`console.log` yerine **kırılma noktası**: Bir değişkenin değerini merak ettiğinizde kodu değiştirip kaydetmek yerine kırılma noktası koyun. **Tüm** değişkenleri aynı anda görürsünüz. Kodda `debugger;` yazmak da aynı işi yapar.

Sık Yapılan DOM Hataları

Hata	Sebebi ve Çözüm
<code>Cannot read properties of null</code> Dinleyici hemen çalışıyor	Element bulunamadı. Seçiciyi kontrol edin; script'i defer ile veya <code></body></code> öncesine koyun.
Form gönderince sayfa yenileniyor	<code>addEventListener("click", fn())</code> — parantezi kaldırın .
Sonradan eklenen öğeler tıklanmıyor	<code>e.preventDefault()</code> eksik.
<code>querySelectorAll</code> sonucunda <code>.map</code> çalışmıyor	Dinleyici eski öğelere bağlı. Olay delegasyonu kullanın.
Stil değişmesine rağmen görüntü aynı	NodeList dizi değil. <code>[...nodeList]</code> ile diziye çevirin.
Değişiklikler yenileyince kayboluyor	CSS özgüllüğü daha yüksek. DevTools Styles panelinde kontrol edin.
	Normal davranış — DOM bellektedir. Kalıcılık için <code>localStorage</code> veya sunucu gerekir.

En sıkısı birincisidir ve çoğu zaman sebebi **script'in DOM'dan önce çalışmasıdır**.

localStorage: Basit Kalıcılık

```
1 // Kaydet - SADECE METİN saklanabilir
2 localStorage.setItem("tema", "koyu");
3 localStorage.setItem("sepet", JSON.stringify(sepetDizisi));
4
5 // Oku
6 const tema = localStorage.getItem("tema"); // "koyu" veya null
7 const sepet = JSON.parse(localStorage.getItem("sepet") ?? "[]");
8
9 // Sil
10 localStorage.removeItem("tema");
11 localStorage.clear(); // hepsini sil
12
13 // Örnek: tema tercihini hatırla
14 const temaButonu = document.querySelector("#tema-butonu");
15 if (localStorage.getItem("tema") === "koyu") {
16     document.body.classList.add("koyu-tema");
17 }
18 temaButonu.addEventListener("click", () => {
19     const koyuMu = document.body.classList.toggle("koyu-tema");
20     localStorage.setItem("tema", koyuMu ? "koyu" : "acik");
21 });
```



Yapay Zeka ile DOM

BÖLÜM 5

DOM Kodunda Yapay Zekanın Riskleri

Sık Ürettiği Sorunlar

- ▶ Kullanıcı verisini `innerHTML` ile basma (XSS)
- ▶ `null` kontrolü yapmama
- ▶ `element.style` ile doğrudan stil verme
- ▶ Her öğeye ayrı dinleyici (delegasyon yerine)
- ▶ `preventDefault()` unutma
- ▶ Erişilebilirlik özneliklerini (`aria-*`) atlama
- ▶ jQuery kalıpları önerme (`$(...)`)

Prompt Kısıtlarınız

Her DOM prompt'una **şu üç satırı** ekleyin: *“Kullanıcı verisi için innerHTML kullanma, textContent kullan.”* *“Element seçiminde null kontrolü yap.”* *“Stil için element.style değil classList kullan.”*

XSS neden önemli? Kullanıcı yorum alanına `<script>...</script>` yazarsa ve siz `innerHTML` ile basarsanız, o kod **tüm ziyaretçilerin tarayıcısında çalışır**.

Yapay Zeka İstemi (Prompt) — DOM Kodu İncelemesi

Asagidaki DOM manipulasyon kodunu denetle.

[KODUNUZU YAPISTIRIN]

Su basliklar altinda incele ve her bulguya [KRITIK] / [ONEMLI] / [ONERI] etiketi ver:

1) GUVENLIK - innerHTML ile kullanıcı verisi basiliyor mu? (XSS) 2) HATA DAYANIKLILIGI - querySelector null donerse ne olur? 3) PERFORMANS - dongude tekrar tekrar DOM'a mi yaziliyor? 4) OLAY YONETIMI - delegasyon kullanilmali miydi? 5) ERISILEBILIRLIK - aria-* oznitelikleri, klavye destegi var mi? 6) AYRIM - stil JS'te mi veriliyor, CSS'te mi olmalı?

Her bulgu icin: sorunun NE OLDUGUNU ve NEDEN sorun oldugunu yaz. Duzeltilmis kodu VERME - sadece degistirilecek satiri "eski -> yeni" olarak goster.

Prompt 2 — XSS Açığını Gösterme

Yapay Zeka İstemi (Prompt) — Güvenlik Deneyi

Asagidaki kod bir yorum formundan gelen metni sayfaya basiyor:

```
const yorum = document.querySelector("#yorum").value; document.querySelector("#yorumlar").innerHTML += "<p>" + yorum + "</p>";
```

GOREV: 1) Bu kodun XSS acigina nasil yol actigini adım adım açıkla. 2) Bir saldirganin yorum kutusuna yazabilecegi ZARARSIZ ama acigi KANITLAYAN bir ornek girdi ver (sadece alert gosteren). 3) Bu girdi calistiginda ne olur? Gercek bir saldirgan bunun yerine ne yapabilirdi? 4) Kodun GUVENLI halini yaz ve neden guvenli oldugunu açıkla. 5) "Kullanici HTML yazabilsin istiyorum" dersem ne yapmaliyim?
Turkce yaz. Egitim amaclı, kendi projemde test edecegim.

Bu deneyi kendi sayfanızda yapın. `alert` çıktığında XSS'i **somut olarak** anlamış olursunuz — ve bir daha `innerHTML` ile kullanıcı verisi basmazsınız.

Prompt 3 — Erişilebilir Bileşen Tasarımı

Yapay Zeka İstemi (Prompt) — Klavye ve Ekran Okuyucu Desteği

Bir mobil "hamburger" menu butonu yaptım:

```
const btn = document.querySelector("#menu-button"); const nav =  
document.querySelector("#main-nav"); btn.addEventListener("click", () =>  
nav.classList.toggle("acik"));
```

GOREV: Bu bileşeni ERISILEBILIR hale getirmek için gereken tüm eklemeleri LISTELE. Her madde için: - Ne eklenmeli (HTML öz nitelik veya JS satırı) - HANGI kullanıcı grubuna fayda sağlar - WCAG olcutu numarası (biliyorsan)

Sunları mutlaka değerlendir: - aria-expanded durumu - aria-controls bağlantısı - Escape tuşu ile kapatma - Menu açılınca odanın ilk linke gitmesi - Menu dışına tiklanınca kapanma - Ekran okuyucunun buton metnini nasıl duyurduğu

KOD VERME. Sadece kontrol listesi olarak yaz. Ben kodu yazacağım, sonra sana denetleteceğim.

Özet ve Değerlendirme

BÖLÜM 6

🔍 Erişim

`querySelector` / `querySelectorAll`.
Bulamazsa `null` döner — kontrol edin.

✍️ Değiştirme

`textContent` (güvenli) vs `innerHTML` (XSS riski). Stil için `classList`, `style` değil.

🖱️ Olaylar

`addEventListener("click", fn)` —
parantezsiz. `e.preventDefault()` form
yenilemesini durdurur. **Delegasyon** ile tek dinleyici.

🤖 Yapay Zeka

`innerHTML` ve `null` kontrolü konusunda
dikkatsiz kod üretir. Prompt'a **güvenlik kısıtları**
ekleyin.

Altın Kural: HTML **yapı**, CSS **görünüm**, JavaScript **durum**. JS'in işi sınıf eklemek/kaldırmaktır — rengi CSS belirler.

Terim	Anlam	Terim	Anlam
DOM	Sayfanın canlı nesne temsili	<code>addEventListener</code>	Olay dinleyici ekler
<code>document</code>	Sayfayı temsil eden nesne	<code>Callback</code>	Olayda çalışan fonksiyon
<code>NodeList</code>	<code>querySelectorAll</code> sonucu	<code>event.target</code>	Olayın olduğu element
<code>textContent</code>	Sadece metin (güvenli)	<code>preventDefault</code>	Varsayılan davranışı durdurur
<code>innerHTML</code>	HTML olarak yorumlar	<code>Bubbling</code>	Olayın yukarı kabarcıklanması
<code>classList</code>	Sınıf yönetimi	<code>Delegasyon</code>	Tek dinleyici ile çoklu yönetim
<code>dataset</code>	<code>data-*</code> öznitelikleri	<code>XSS</code>	Zararlı kod enjeksiyonu
<code>createElement</code>	Yeni element oluşturur	<code>closest()</code>	En yakın atayı bulur

1 `textContent` ile `innerHTML` arasındaki fark nedir? Hangi durumda hangisini kullanmalısınız ve neden?

2 Aşağıdaki kod neden çalışmaz? İki farklı sebep yazınız:

```
const btn = document.querySelector("menu-button");  
btn.addEventListener("click", menuAc());
```

3 Olay delegasyonu nedir? Hangi **iki** problemi çözer?

4 `e.target` ile `e.currentTarget` arasındaki fark nedir?

5 Bir form gönderiminde `preventDefault()` çağırmasanız ne olur?

6 `classList.toggle("acik")` ifadesi ne döndürür? Bu dönüş değerini nerede kullanabilirsiniz?

7 Sayfayı yenilediğinizde JavaScript ile yaptığınız değişiklikler neden kaybolur?

Laboratuvar Föyü 12

- ▶ Seçici ve erişim alıştırımları
- ▶ `textContent` vs `innerHTML` deneyi
- ▶ Mobil menünün tamamlanması (HTML+CSS+JS)
- ▶ Katalog verisinden **dinamik kart çizimi**
- ▶ Canlı arama ve filtreleme
- ▶ Form doğrulama
- ▶ Olay delegasyonu uygulaması
- ▶ **XSS deneyi** (kendi sayfanızda)

Hazırlık

Okuma: Kitap Ünite 2.3 — “DOM Manipülasyonu”, “Olaylar” **Kaynaklar:**

- ▶ developer.mozilla.org/docs/Web/API/Document
- ▶ javascript.info/document
- ▶ owasp.org — XSS

Gelecek hafta: **Asenkron JavaScript** — sunucudan veri çekeceğiz.

Haftaya Görüşmek Üzere!

Gelecek Hafta:

Asenkron JavaScript, Fetch API ve JSON

Senkron/asenkron farkı ▪ ▪ ▪ / ▪ JSON veri formatı