Insider | USEINSIDER.COM

CodeCraft
Bootcamp'25

Javascript

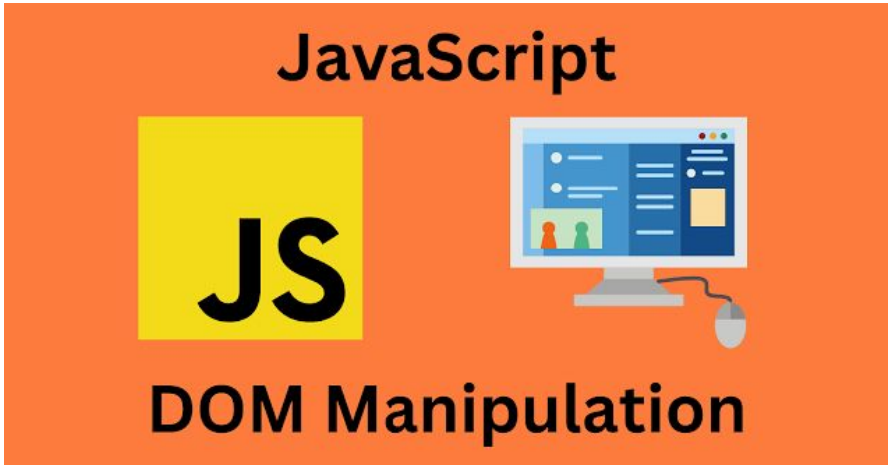
Creating, Removing, and Updating Nodes, Event Listeners & User Interactions

DOM Manipulation

Element Oluşturma
- document.createElement()
- appendChild() ve insertBefore()
- innerHTML

Element Silme
- removeChild()
- innerHTML = “”

Element Güncelleme
- textContent, innerHTML ve innerText
- setAttribute() ve classList.add()
- replaceChild()



Event Listeners

Temel Kullanım

- addEventListener(event, function)
- removeEventListener(event, function)
- onclick Attribute vs addEventListener
  - addEventListener() ile bir elemana birden fazla event listener ekleyebilirsin.
  - onclick ise sadece bir tane işlev alır; yeni bir değer atanırsa eskisi silinir.

Frequently used events

- **click**
  - Bir kullanıcı bir elemana (buton, link, div vb.) tıkladığında çalışır.
- **mouseover**:
  - Kullanıcının bir elementin üzerine gelmesiyle çalışır.
  - Genellikle hover efektleri oluşturmak için kullanılır.
- **mouseout**
  - Kullanıcı bir elementin üzerinden ayrıldığında tetiklenir.
  - mouseover olayının tersidir.

Frequently used events

- **keydown**
  - Kullanıcı bir tuşa bastığında çalışır.
  - Sürekli basılı tutulursa, tekrar tekrar tetiklenir.
- **keyup**
  - Kullanıcı bir tuşa bastıktan sonra bıraktığında çalışır.
  - keydown olayından farklı olarak, bir tuşun gerçekten bırakıldığı anı yakalar.
- **input**
  - Kullanıcı bir input, textarea veya select gibi bir form elemanında değişiklik yaptığında çalışır.
  - keyup ve change olaylarının birleşimi gibidir.

Event Object and Properties

- **event.target**
  - Kullanıcının etkileşime geçtiği gerçek HTML elementini döndürür.
  - event.target, hangi elementin olayı tetiklediğini öğrenmek için kullanılır.

- **event.currentTarget**
  - Olayın bağlandığı element
  - Olayın hangi element üzerinde dinlendiğini öğrenmek

- **event.type**
  - Gerçekleşen olayın tipini döndürür.
  - click, mouseover, keydown gibi olayları kontrol etmek için kullanılır.

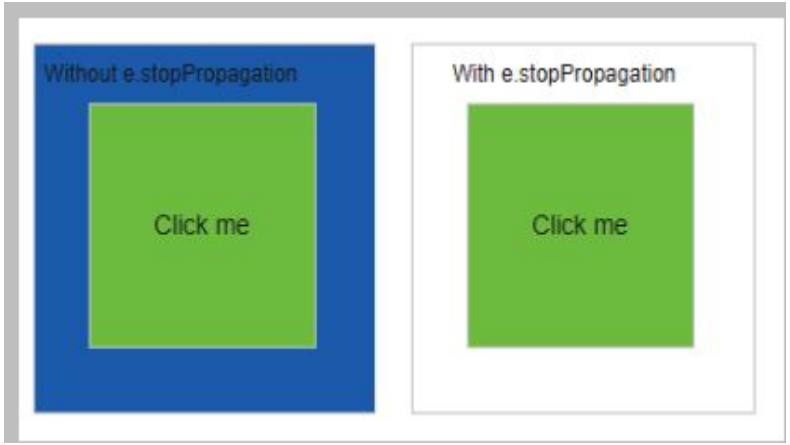
event.preventDefault()

Bazı HTML elementleri varsayılan bir davranışa sahiptir. Örneğin:

- Link (<a>) tıklandığında sayfa yönlendirilir.
- Form (<form>) gönderildiğinde sayfa yenilenir.
- <input type="checkbox"> seçili durumu değiştirir.
- event.preventDefault() bu varsayılan işlemleri engeller.

event.stopPropagation()

- HTML’de iç içe geçmiş elementlerde, eventler otomatik olarak yukarı doğru yayılır (Bubbling).
- İç içe elementlerde tıklama gibi olayların yayılmasını durdurmak için kullanırız. Yani event.stopPropagation() kullanıldığında, olay üst elementlere gitmez.



Event Delegation

Event Delegation, birden fazla elemente ayrı ayrı event listener eklemek yerine, ebeveyn (parent) elemente tek bir event listener ekleyerek olayları yönetme tekniğidir.

Örneğin: Bir <ul> içinde birçok <li> varsa, her <li> için ayrı event listener eklemek yerine, olayları <ul> üzerinden dinleyebiliriz.

Performans Avantajları

Daha az bellek kullanımı: Tek bir event listener kullanılır.
Dinamik öğelerle çalışır: Sonradan eklenen elementler de otomatik olarak event listener'dan etkilenir.
Daha temiz ve yönetilebilir kod: Tek bir noktadan kontrol sağlanır.

event.target ile Çalışmak

event.target, olayın gerçekleştiği spesifik elementi döndürür.
Event Delegation kullanırken, hangi elementin etkilendiğini event.target ile belirleyebiliriz.

Form Controls and Interactions

Form elemanları (input, select, checkbox vb.) ile yapılan etkileşimleri yönetmek için input, change, submit gibi olaylar kullanılır.

input ve change Olayları

input olayı: Kullanıcı her karakter girdiğinde tetiklenir (canlı değişiklikler için kullanılır).
change olayı: Kullanıcı bir değeri değiştirip, odak kaybettiğinde (blur) çalışır.

Form Doğrulama

- Client-side validation (İstemci tarafı doğrulama), kullanıcı yanlış giriş yaptığında anlık geri bildirim sağlamak için kullanılır.
- required, minlength, pattern gibi HTML özellikleriyle veya JavaScript ile kontrol edilebilir.
- event.preventDefault() kullanılarak yanlış form gönderiminin önüne geçilebilir.

Event Bubbling & Capturing

Bir olay meydana geldiğinde, tarayıcı bunu iki aşamada işler:

1. **Capturing (Yakalama Aşaması) → Dışarıdan içeriye** doğru ilerler.
2. **Bubbling (Kabarcıklanma Aşaması) → İçeriden dışarıya** doğru ilerler.

Varsayılan olarak, olaylar **bubbling** aşamasında çalışır.

`stopPropagation()` ve `capture` Kullanımı

- `stopPropagation()`: Olayın üst elementlere yayılmasını engeller.
- `addEventListener(event, handler, true)`: Üçüncü parametre `true` olduğunda, olay **capturing aşamasında** çalışır (bubbling yerine).

Gerçek Zamanlı Görev Yönetim Uygulaması

1 - Görev Listesi Arayüzü (HTML & CSS & JS ile Dinamik İçerik)

- Sayfa yüklendiğinde, **boş bir görev listesi** gösterilecek.
- Kullanıcı, bir görev ekleyebilecek.
- Her görev için:
  - **Başlık** (Zorunlu Alan)
  - **Açıklama** (Opsiyonel)
  - **Öncelik** (Düşük / Orta / Yüksek - Radio butonlarıyla)
  - **Tamamlandı mı?** (Varsayılan olarak tamamlanmamış olacak)

2 - Görev Ekleme & DOM Manipülasyonu

- Kullanıcı formu doldurup **"Ekle" butonuna** bastığında, yeni görev listeye eklenecek.
- Görev **dinamik olarak HTML'e eklenmeli**, yani sayfa yenilendiğinde sıfırlanmalı.
- Form gönderildikten sonra **form inputları temizlenmeli.**

Gerçek Zamanlı Görev Yönetim Uygulaması

3 - Olay Yönetimi & Event Delegation

- Kullanıcı, **her görev satırında** bulunan butonları kullanarak:
  - **Tamamlandı olarak işaretleyebilir.** (Yeşil bir arka plan eklenebilir)
  - Görevi listeden silebilir.
- event.target kullanarak dinamik elemanları dinlemek gerekiyor.
- stopPropagation() ile istenmeyen event bubbling önlenmeli.

4 - Form Doğrulama ve Hata Yönetimi

- Kullanıcı, **boş bir görev ekleyemez.**
- **Öncelik seçilmezse**, bir hata mesajı gösterilmeli.
- try-catch bloğu ile beklenmedik hatalar yakalanmalı.

5 - Filtreleme & Sıralama (Opsiyonel)

- Kullanıcı, **"Sadece tamamlananları göster"** butonuna basınca tamamlanan görevleri filtreleyebilmeli.
- **Önceliğe göre sıralama** seçeneği eklenebilir (Düşük → Yüksek).