# ChatGPT

Tabii! Bu React bileşeni ( `MouseEvents` ), fareyle etkileşimli olayları (mouse events) izlemek için yazılmış bir örnektir. Kodda `onMouseMove` , `onMouseOver` , `onMouseEnter` , ve `onDoubleClick` gibi olaylar kullanılıyor ve bu olayların her biri belirli fare hareketlerine veya tıklamalara tepki verir.

## Kodun Genel Yapısı

1. **React ve useState Kullanımı:**
   - `useState` React hook'u, bileşenin durumunu yönetmek için kullanılır. Burada fare konumlarını (X ve Y koordinatları) ve fare öğesine gelip gitme durumunu ( `visible` ) saklamak için kullanılıyor.
2. **Olayları İşlemek:**
   - Farklı fare olaylarını (mousemove, mouseover, mouseenter, double click) işlemek için fonksiyonlar tanımlanmış.
   - Bu olaylar gerçekleştiğinde, state güncellenir ve UI (kullanıcı arayüzü) yeniden render edilir.

## Bileşen Yapısı

1. **State Değişkenleri:**

```javascript
const [visible, setVisible] = useState(false);
const [coordX, setCoordX] = useState("");
const [coordY, setCoordY] = useState("");
```

   - `visible` : Bu state, fare bir öğenin üzerine geldiğinde ekranda `Mouse üzerimde` yazısını göstermek için kullanılır.
   - `coordX` ve `coordY` : Bu state'ler fare hareketinin koordinatlarını tutar. Fare hareket ettikçe bu değerler güncellenir.
2. `handleDoubleClick` **Fonksiyonu:**

```javascript
const handleDoubleClick = (e) => e.target.classList.toggle("bg-danger");
```

- `onDoubleClick` olayını işleyen bu fonksiyon, fare iki kez tıklanmasıyla tetiklenir.
- `e.target.classList.toggle("bg-danger")` ile, tıklanan öğenin CSS sınıfı değiştirilir. Bu, öğeye `bg-danger` sınıfı ekler veya çıkarır. Bu sınıf, genellikle kırmızı bir arka plan rengi (Bootstrap'ta) için kullanılır.

3. `handleMouseMove` **Fonksiyonu:**

```javascript
const handleMouseMove = (e) => {
  setCoordX(e.nativeEvent.offsetX);
  setCoordY(e.nativeEvent.offsetY);
};
```

- Bu fonksiyon, fare bir öğe üzerinde hareket ettikçe tetiklenir. `e.nativeEvent.offsetX` ve `e.nativeEvent.offsetY` ile fare pozisyonunun öğe içindeki göreceli koordinatları alınır ve state güncellenir. Bu sayede, fare hareket ettikçe X ve Y koordinatları ekranda görüntülenir.

4. **HTML Yapısı:**

```jsx
<div className="container text-center d-flex flex-column align-items-center mt-4">
```

- Bu bir container div, içinde metinler ve fare hareketlerine bağlı öğeler var. `text-center`, `d-flex`, `align-items-center`, `mt-4` gibi sınıflar, Bootstrap kütüphanesinden gelen stil sınıflarıdır.
- `onMouseOver` ve `onMouseLeave` olaylarını kullanan bir div:

```jsx
<div
  onMouseOver={() => setVisible(true)}
  onMouseLeave={() => setVisible(false)}
  id="todo-1"
  className="bg-success text-light w-50 p-4"
>
  onMouseOver & onMouseLeave
</div>
```

- `onMouseOver`: Fare, bu öğeye geldiğinde `setVisible(true)` ile `visible` state'i `true` yapılır ve "Mouse üzerimde" mesajı gösterilir.
- `onMouseLeave`: Fare bu öğeden ayrıldığında `setVisible(false)` ile "Mouse üzerimde" mesajı kaybolur.

- `onDoubleClick` olayını kullanan bir div:

```jsx
<div
  onDoubleClick={handleDoubleClick}
  id="todo-2"
  className="bg-success text-light w-50 p-4 mt-4"
>
  onDoubleClick
</div>
```

  - Fare iki kez tıklandığında `handleDoubleClick` fonksiyonu çalışır ve öğeye `bg-danger` sınıfı eklenir veya çıkarılır.

- `onMouseMove`, `onMouseOver`, ve `onMouseEnter` olaylarını kullanan diğer div'ler:

```jsx
<div
  onMouseMove={handleMouseMove}
  id="todo-3"
  className="bg-success text-light w-50 p-4 my-4"
>
  <span>onMouseMove</span>
</div>
<div
  onMouseOver={handleMouseMove}
  id="todo-4"
  className="bg-success text-light w-50 p-4 my-4"
>
  <span>onMouseOver</span>
</div>
<div
  onMouseEnter={handleMouseMove}
  id="todo-5"
  className="bg-success text-light w-50 p-4 my-4"
>
```

```
  <span>onMouseEnter</span>
</div>
```

- Bu üç div, fare hareketlerini (move, over, enter) takip etmek için kullanılır.
- `onMouseMove`: Fare öğenin üzerinde hareket ettikçe tetiklenir.
- `onMouseOver`: Fare öğenin üzerine geldiğinde tetiklenir, ancak sadece öğenin üzerine geldiği andan itibaren etkinleşir, içindeki çocuk öğelere de bu etki geçer.
- `onMouseEnter`: Fare sadece öğenin üzerine girdiğinde tetiklenir, çocuk öğeler etkilemez.

5. **Koordinatların Görüntülenmesi:**

```jsx
X: {coordX} and Y:{coordY}
```

- Fare hareket ettikçe `coordX` ve `coordY` state'leri güncellenir ve bu değerler JSX içinde ekrana yazdırılır.

## Özetle:

- `onMouseOver`, `onMouseEnter` **ve** `onMouseLeave`**:** Fare etkileşimlerini (gelme, gitme) izler.
- `onMouseMove`**:** Fare hareketini takip eder.
- `onDoubleClick`**:** Fare iki kez tıklanmasıyla öğeye `bg-danger` sınıfını ekler veya çıkarır.
- **Koordinatlar:** Fare hareket ettikçe, öğe içindeki X ve Y koordinatları güncellenir ve UI'de gösterilir.

Bu kod, fare etkileşimlerini takip etmek ve bu etkileşimlere göre kullanıcı arayüzünü dinamik olarak güncellemek için iyi bir örnektir.