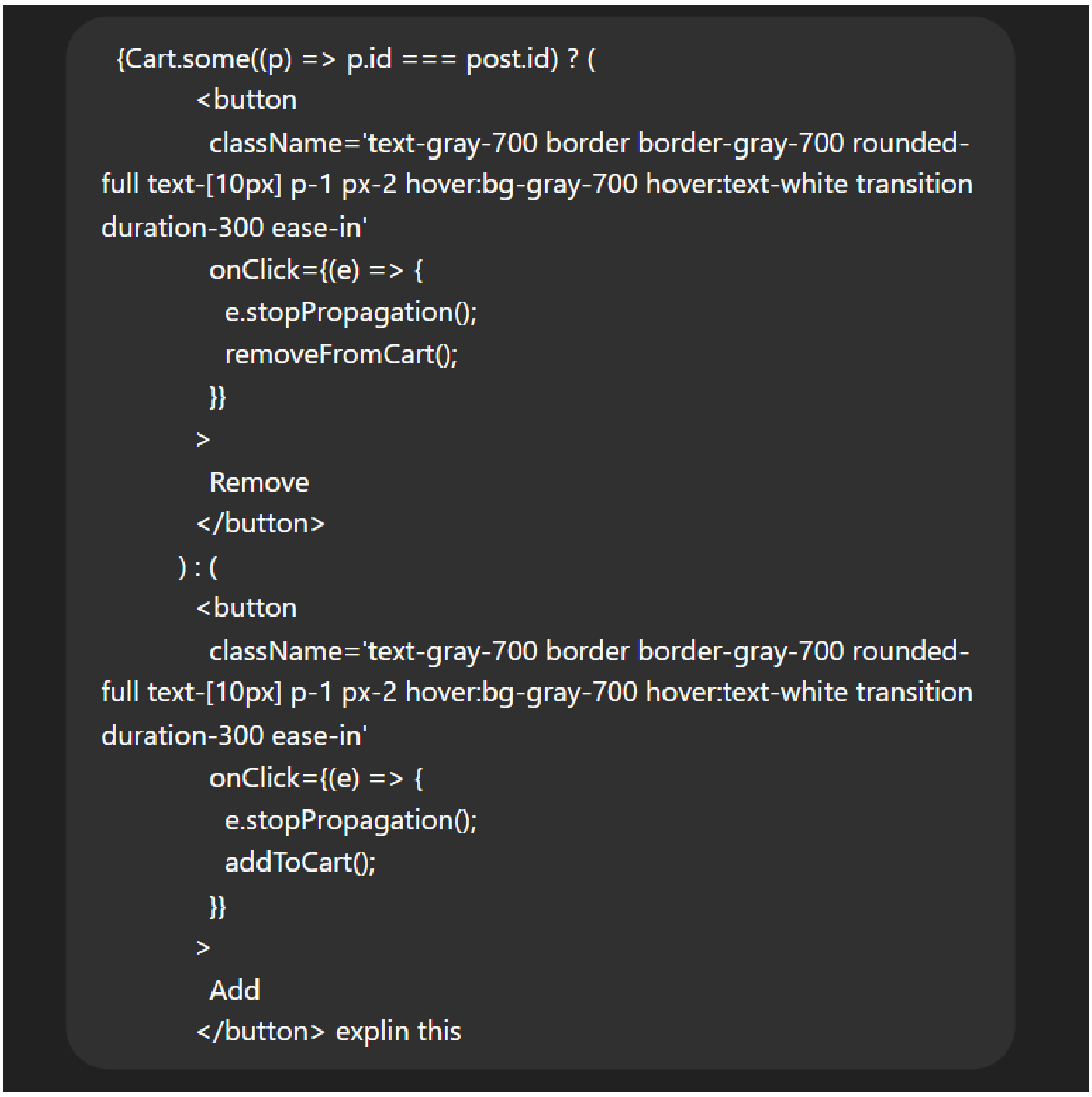
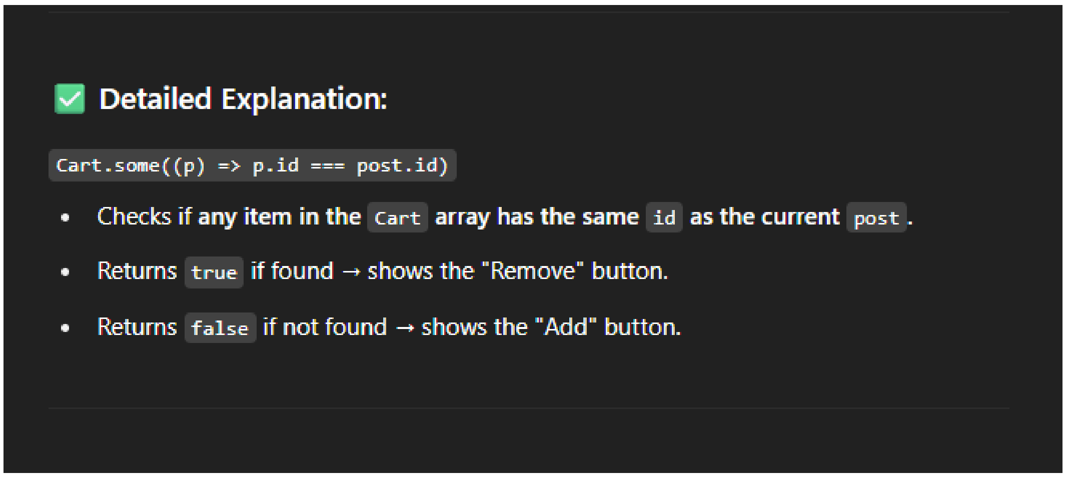


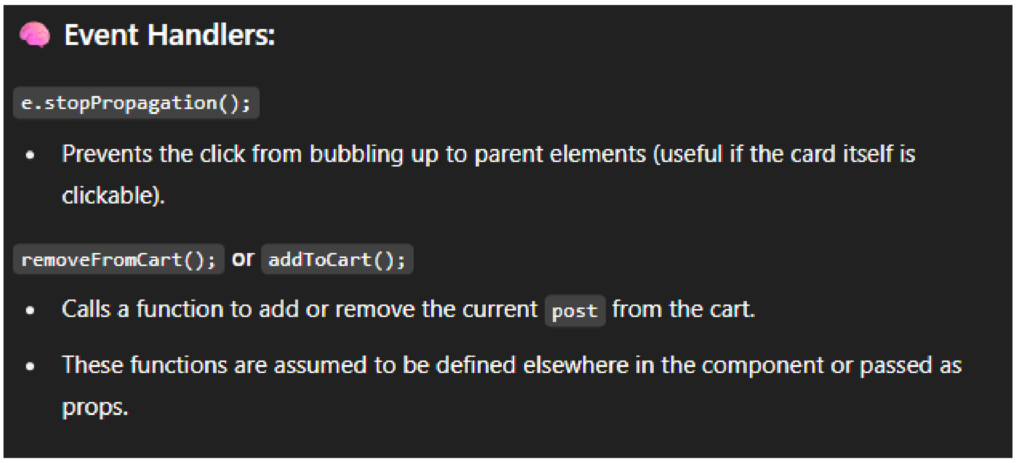
```
 {Cart.some((p) => p.id === post.id) ? (
          <button
            className='text-gray-700 border border-gray-700 rounded-
full text-[10px] p-1 px-2 hover:bg-gray-700 hover:text-white transition
duration-300 ease-in'
            onClick={(e) => {
              e.stopPropagation();
              removeFromCart();
            }}
          >
            Remove
          </button>
        ) : (
          <button
            className='text-gray-700 border border-gray-700 rounded-
full text-[10px] p-1 px-2 hover:bg-gray-700 hover:text-white transition
duration-300 ease-in'
            onClick={(e) => {
              e.stopPropagation();
              addToCart();
            }}
          >
            Add
          </button> explin this
```



✅ Detailed Explanation:

`Cart.some((p) => p.id === post.id)`

- Checks if **any item in the** `Cart` **array has the same** `id` **as the current** `post`.
- Returns `true` if found → shows the "Remove" button.
- Returns `false` if not found → shows the "Add" button.



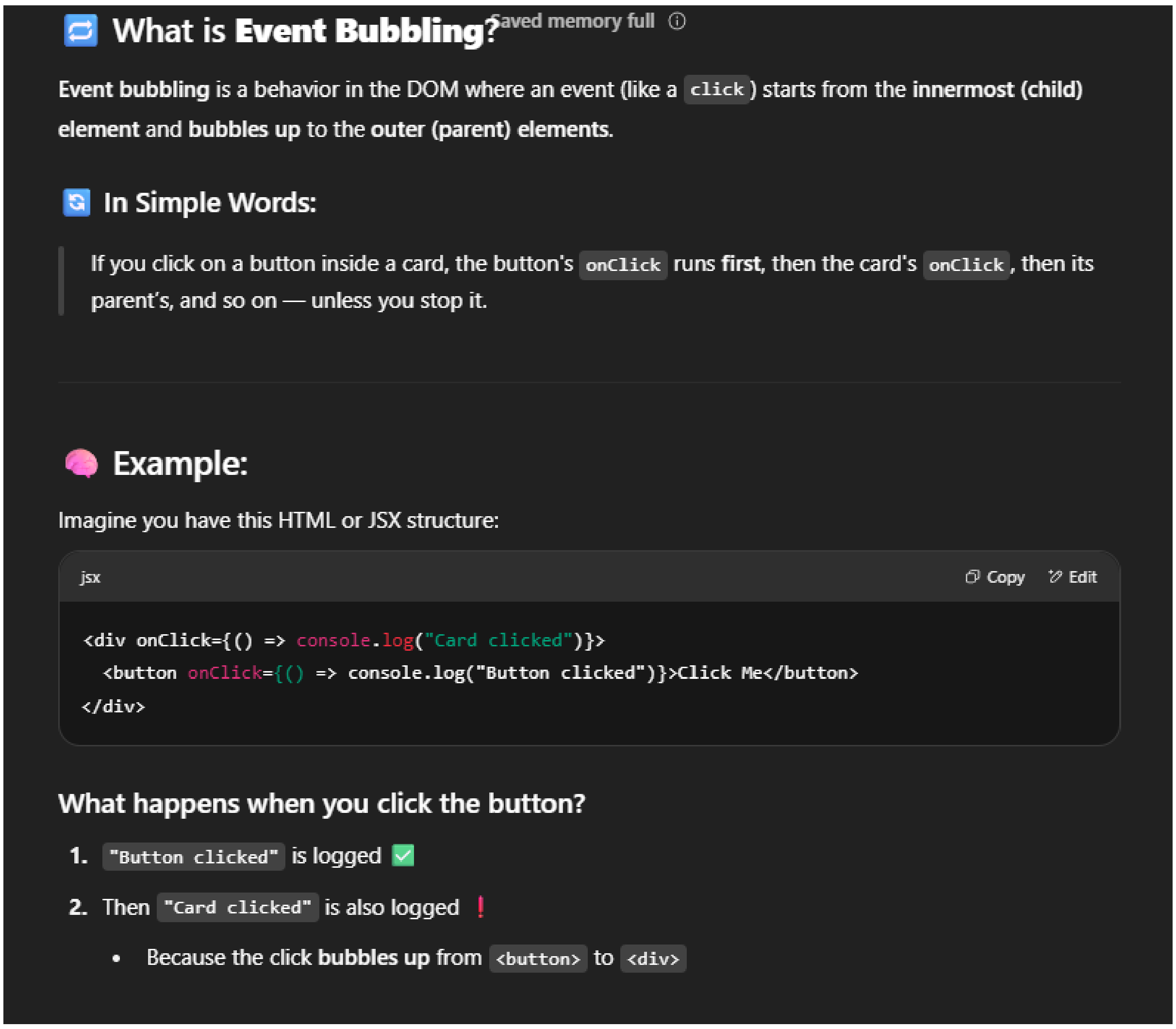
🧠 Event Handlers:

`e.stopPropagation();`

- Prevents the click from bubbling up to parent elements (useful if the card itself is clickable).

`removeFromCart();` or `addToCart();`

- Calls a function to add or remove the current `post` from the cart.
- These functions are assumed to be defined elsewhere in the component or passed as props.



🔁 What is **Event Bubbling?**

Event bubbling is a behavior in the DOM where an event (like a `click`) starts from the **innermost (child) element** and **bubbles up** to the **outer (parent) elements**.

🔄 In Simple Words:

> If you click on a button inside a card, the button's `onClick` runs **first**, then the card's `onClick`, then its parent's, and so on — unless you stop it.

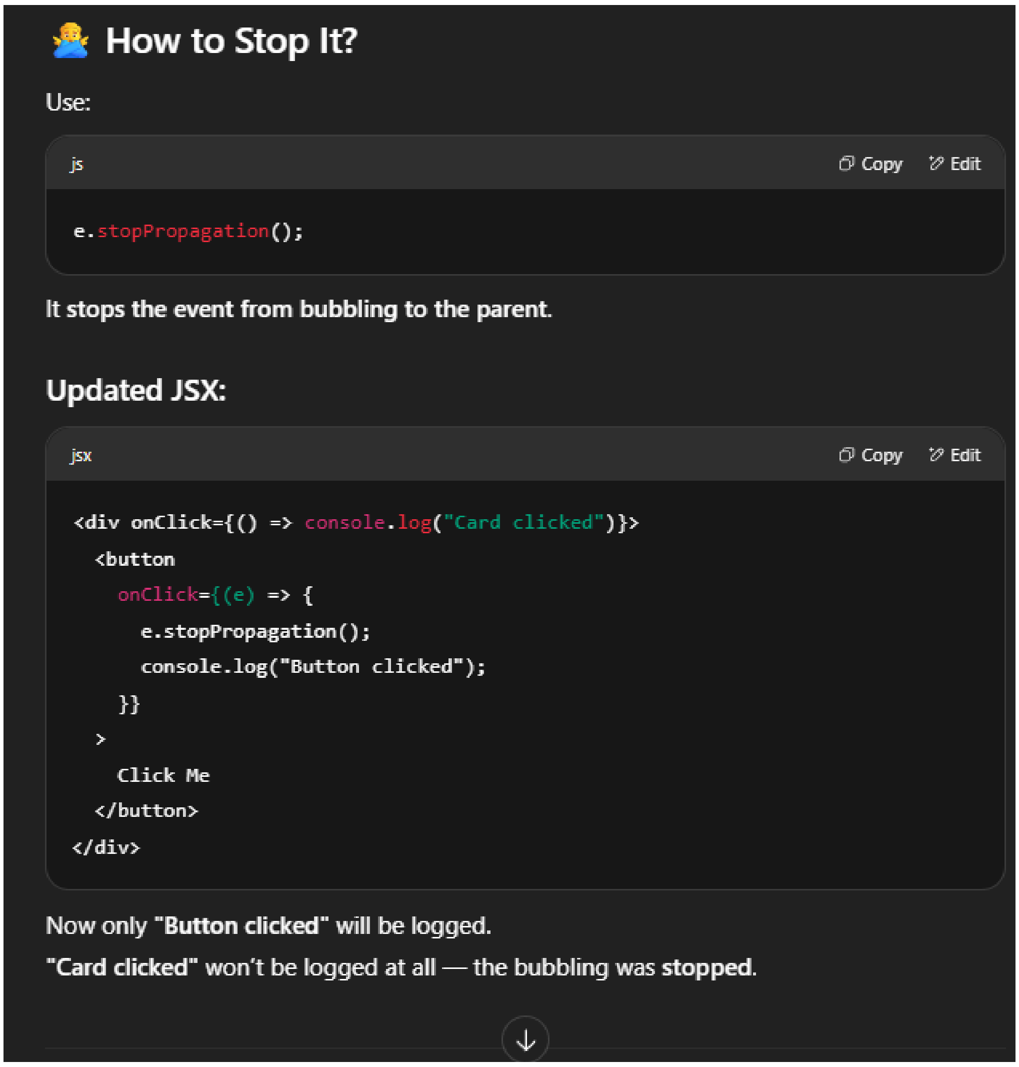
🧠 Example:

Imagine you have this HTML or JSX structure:

```jsx
<div onClick={() => console.log("Card clicked")}>
  <button onClick={() => console.log("Button clicked")}>Click Me</button>
</div>
```

What happens when you click the button?

1. `"Button clicked"` is logged ✅
2. Then `"Card clicked"` is also logged ❗
   - Because the click **bubbles up** from `<button>` to `<div>`



🙅 How to Stop It?

Use:

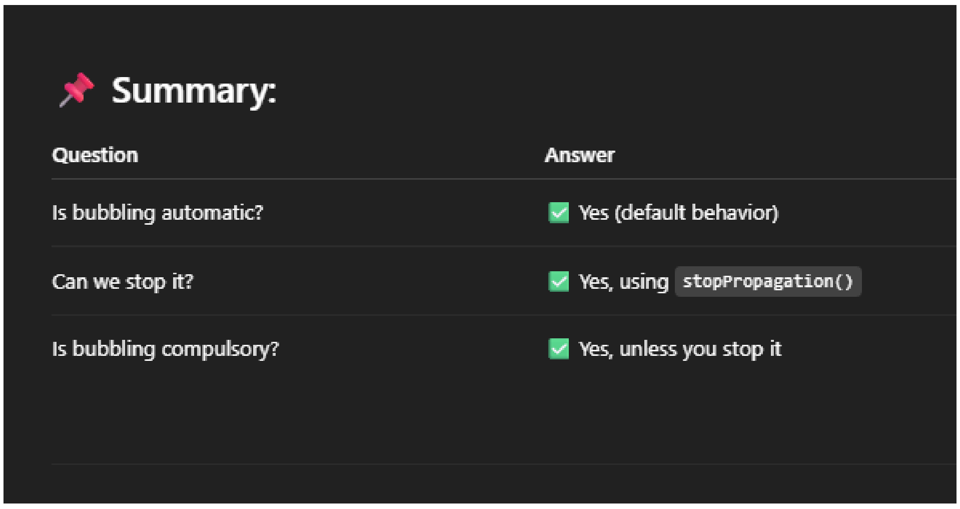
```js
e.stopPropagation();
```

It stops the event from bubbling to the parent.

Updated JSX:

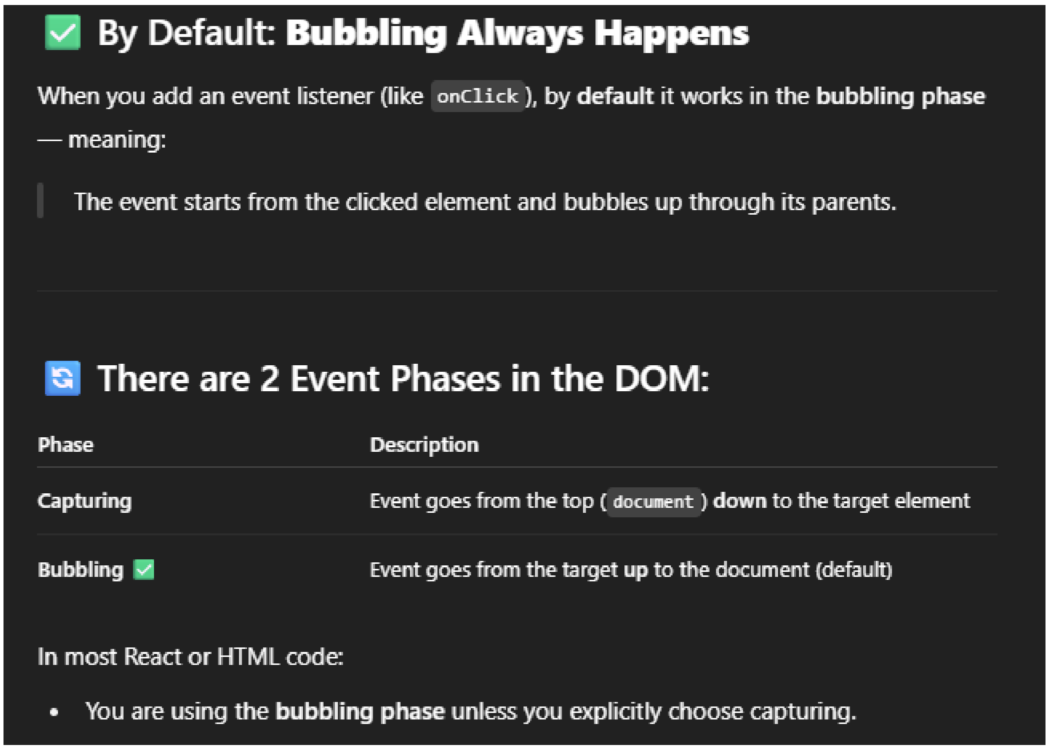
```jsx
<div onClick={() => console.log("Card clicked")}>
  <button
    onClick={(e) => {
      e.stopPropagation();
      console.log("Button clicked");
    }}
  >
    Click Me
  </button>
</div>
```

Now only **"Button clicked"** will be logged.
"Card clicked" won't be logged at all — the bubbling was **stopped**.



📌 Summary:

Question	Answer
Is bubbling automatic?	✅ Yes (default behavior)
Can we stop it?	✅ Yes, using `stopPropagation()`
Is bubbling compulsory?	✅ Yes, unless you stop it



✅ By Default: **Bubbling Always Happens**

When you add an event listener (like `onClick`), by **default** it works in the **bubbling phase** — meaning:

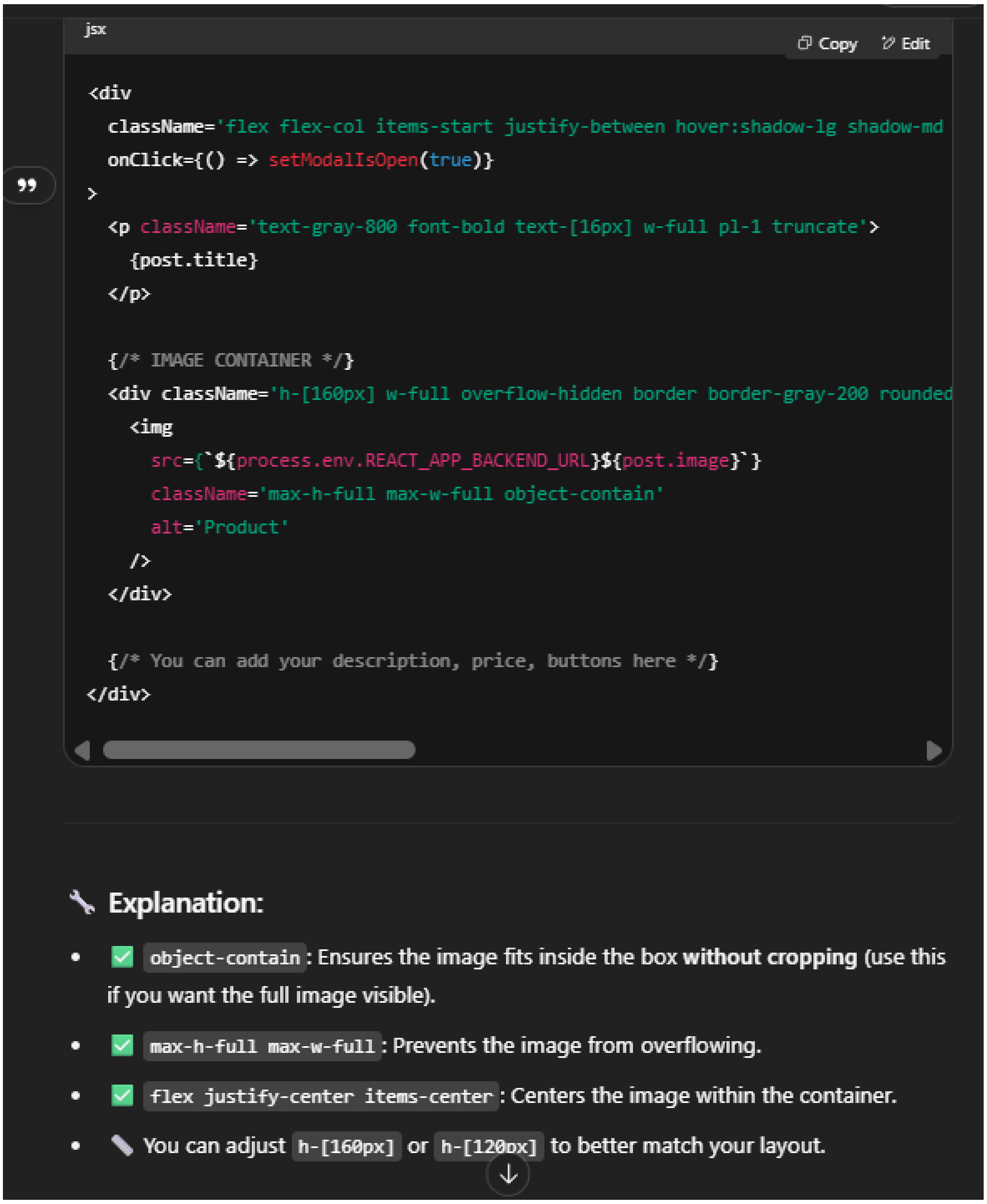
> The event starts from the clicked element and bubbles up through its parents.

🔄 There are 2 Event Phases in the DOM:

Phase	Description
Capturing	Event goes from the top (`document`) **down** to the target element
Bubbling ✅	Event goes from the target **up** to the document (default)

In most React or HTML code:

- You are using the **bubbling phase** unless you explicitly choose capturing.


jsx

Copy Edit

```jsx
<div
  className='flex flex-col items-start justify-between hover:shadow-lg shadow-md
  onClick={() => setModalIsOpen(true)}
>
  <p className='text-gray-800 font-bold text-[16px] w-full pl-1 truncate'>
    {post.title}
  </p>

  {/* IMAGE CONTAINER */}
  <div className='h-[160px] w-full overflow-hidden border border-gray-200 rounded
    <img
      src={`${process.env.REACT_APP_BACKEND_URL}${post.image}`}
      className='max-h-full max-w-full object-contain'
      alt='Product'
    />
  </div>

  {/* You can add your description, price, buttons here */}
</div>
```

🔧 Explanation:

- ✅ `object-contain`: Ensures the image fits inside the box **without cropping** (use this if you want the full image visible).
- ✅ `max-h-full max-w-full`: Prevents the image from overflowing.
- ✅ `flex justify-center items-center`: Centers the image within the container.
- 📏 You can adjust `h-[160px]` or `h-[120px]` to better match your layout.

✅ **object-contain**

📌 **What it does:**

It makes sure the **whole image is shown inside** the container **without cutting or cropping**, **even if it leaves blank spaces**.

📷 **Visual Example:**

- Like when you fit a photo inside a photo frame **without trimming** it.

📌 **Use case:**

If your product images have different dimensions (some are tall, some are wide), and you want to **see the entire image always**, use object-contain.

✅ **max-h-full max-w-full**

📌 **What it does:**

- max-h-full: Prevents the image height from exceeding the container.
- max-w-full: Prevents the image width from overflowing the container.

📷 **Visual Example:**

- Imagine an image sitting **within its box**—it will **never stretch** beyond it.

💡 **Tip:**

You apply these to the <img> tag inside a container like this:

```
<div className="h-[160px] w-full flex justify-center items-center">
  <img src="..." className="object-contain max-h-full max-w-full" />
</div>
```

✅ **flex justify-center items-center**

📌 **What it does:**

It centers the image **both vertically and horizontally** inside its container.

📷 **Visual Example:**

- Think of putting a small image in a big frame and keeping it **perfectly centered**.

html

```
<div className="flex justify-center items-center h-[160px] w-full">
  <img src="..." className="..." />
```

</div>

📏 **Adjusting h-[160px] or h-[120px]**

📌 **What it does:**

- This is the **height of the image container**.
- Increasing it (e.g., h-[160px]) makes the image area taller.
- Decreasing it (e.g., h-[120px]) makes it shorter.

🔧 **How to choose?**

Try different values based on how much space your card layout allows. Example:

<div className="h-[140px] w-full ..."> // try 120px, 140px, 160px, etc.