Event Handling
Adding Event Listeners:
javascript
Copy code
button.addEventListener('click', () => {
  alert('Button clicked!');
});
Event Object: Provides information about the event, e.g., event.target to get the element that triggered the event.

In JavaScript, event handling is a key feature for interacting with user actions or other dynamic occurrences. You can handle events such as clicks, mouse movements, form submissions, and more using event listeners. Here's a breakdown:

1. Adding Event Listeners

An event listener allows you to run a function when a specific event occurs on an element.

javascript

```javascript
const button = document.getElementById('myButton');
button.addEventListener('click', () => {
  alert('Button clicked!');
});
```

- `addEventListener('event', callbackFunction)`:
  - `'click'`: The type of event you're listening for (click, in this case).
  - `callbackFunction`: A function that executes when the event occurs.

2. Event Object

When an event occurs, an `event` object is passed to the event listener. It contains details about the event, such as the type of event and the target element that triggered it.

Example using the event object:

javascript

```javascript
button.addEventListener('click', (event) => {
  console.log('Button clicked:', event.target); // Logs the button element
});
```

- `event.target`: Returns the element that triggered the event (in this case, the button).
- **Other properties in the `event` object:**
  - `event.type`: The type of event (e.g., "click").
  - `event.preventDefault()`: Prevents the default action associated with the event.
  - `event.stopPropagation()`: Stops the event from bubbling up to parent elements.

3. Removing Event Listeners

You can remove an event listener if needed using `removeEventListener`.

javascript

```javascript
function handleClick() {
  alert('Button clicked!');
}

button.addEventListener('click', handleClick);

// To remove the event listener:
button.removeEventListener('click', handleClick);
```

This is useful when you want to stop responding to events dynamically.

By utilizing event listeners, you can create interactive and dynamic web applications that respond to user inputs or other events in real time.