Eventi e Interattività

Gli **eventi e l'interattività** sono aspetti fondamentali dello sviluppo web. Gli eventi permettono di reagire alle azioni dell'utente, come cliccare un pulsante, digitare sulla tastiera, o spostare il mouse. JavaScript è il linguaggio principale per la gestione degli eventi, rendendo le pagine web interattive e dinamiche.

1. Introduzione agli eventi

Gli **eventi** sono azioni o accadimenti che avvengono nel browser, come un clic del mouse o la pressione di un tasto. JavaScript permette di "ascoltare" questi eventi e di eseguire del codice in risposta.

Esempi di eventi comuni

1. **click**: Quando l'utente clicca su un elemento.
2. **keypress**: Quando l'utente preme un tasto sulla tastiera (in disuso, sostituito da `keydown` e `keyup`).
3. **mouseover**: Quando il puntatore del mouse passa sopra un elemento.

2. La funzione `addEventListener`

`addEventListener` è il metodo principale per collegare un gestore (handler) a un evento. È preferibile rispetto all'uso diretto delle proprietà degli eventi (ad esempio `onclick`), perché consente:

- Maggior flessibilità.
- Aggiunta di più gestori per lo stesso evento.
- Rimozione di gestori quando non sono più necessari.

Sintassi

```
element.addEventListener(event, function, useCapture);
```

- **`event`**: Tipo di evento da ascoltare (ad esempio, "click", "keypress").
- **`function`**: Funzione da eseguire quando si verifica l'evento.
- **`useCapture`**: Booleano facoltativo, usato per specificare se si vuole catturare l'evento nella fase di cattura (fase avanzata).

3. Esempi pratici

Esempio 1: Evento `click` su un bottone

Codice HTML:

```
<button id="myButton">Cliccami!</button>
<p id="output"></p>
```

Codice JavaScript:

```
// Selezioniamo il bottone e il paragrafo
```

```
const button = document.getElementById("myButton");
const output = document.getElementById("output");

// Aggiungiamo un evento "click"
button.addEventListener("click", function () {
    output.textContent = "Hai cliccato il bottone!";
});
```

Esempio pratico: Quando l'utente clicca sul bottone, nel paragrafo appare il messaggio "Hai cliccato il bottone!".

Esempio 2: Evento `keypress` per rilevare l'inserimento di un testo

Codice HTML:

```
<input type="text" id="textInput" placeholder="Scrivi qui">
<p id="keyOutput"></p>
```

Codice JavaScript:

```
// Selezioniamo il campo di input e il paragrafo
const textInput = document.getElementById("textInput");
const keyOutput = document.getElementById("keyOutput");

// Aggiungiamo un evento "keydown" per monitorare i tasti premuti
textInput.addEventListener("keydown", function (event) {
    keyOutput.textContent = `Hai premuto il tasto: ${event.key}`;
});
```

Esempio pratico: Quando l'utente digita un carattere nel campo di input, viene mostrato quale tasto è stato premuto.

Esempio 3: Evento `mouseover` per cambiare stile

Codice HTML:

```
<div id="hoverBox" style="width: 200px; height: 200px; background-color:
lightblue;">
    Passa il mouse qui
</div>
```

Codice JavaScript:

```
// Selezioniamo il box
const hoverBox = document.getElementById("hoverBox");

// Aggiungiamo un evento "mouseover" per cambiare colore
hoverBox.addEventListener("mouseover", function () {
    hoverBox.style.backgroundColor = "orange";
});

// Aggiungiamo un evento "mouseout" per ripristinare il colore originale
hoverBox.addEventListener("mouseout", function () {
    hoverBox.style.backgroundColor = "lightblue";
});
```

Esempio pratico: Quando il mouse passa sopra il box, il colore cambia in arancione; quando il mouse esce, torna al colore originale.

4. Uso di funzioni esterne

Invece di scrivere funzioni direttamente dentro `addEventListener`, possiamo utilizzare funzioni definite esternamente per migliorare leggibilità e riutilizzabilità.

Codice JavaScript:

```
function showAlert() {
    alert("Bottone cliccato!");
}

document.getElementById("myButton").addEventListener("click", showAlert);
```

5. Rimuovere un gestore eventi

Puoi rimuovere un gestore eventi con il metodo `removeEventListener`.

Esempio:

```
function logMessage() {
    console.log("Questo messaggio compare solo una volta.");
    button.removeEventListener("click", logMessage);
}

const button = document.getElementById("myButton");
button.addEventListener("click", logMessage);
```

Spiegazione: Dopo il primo clic, il gestore viene rimosso e il messaggio non appare più.

6. Event Delegation (Gestione ottimizzata degli eventi)

Se hai molti elementi dinamici, puoi utilizzare il principio di **delegazione degli eventi**, dove un singolo gestore "ascolta" gli eventi su un elemento genitore.

Esempio: Codice HTML:

```
<ul id="itemList">
    <li>Elemento 1</li>
    <li>Elemento 2</li>
    <li>Elemento 3</li>
</ul>
```

Codice JavaScript:

```
const itemList = document.getElementById("itemList");

// Aggiungiamo un unico evento "click" all'elemento padre
itemList.addEventListener("click", function (event) {
    if (event.target.tagName === "LI") {
        alert(`Hai cliccato su: ${event.target.textContent}`);
    }
```

```
});
```

Spiegazione: Anche se aggiungiamo nuovi `<li>` dinamicamente, il gestore eventi funziona perché è legato al `<ul>`.

Conclusione

Gli eventi e l'interattività sono essenziali per creare esperienze utente coinvolgenti. Grazie a `addEventListener` e alla gestione ottimizzata, possiamo scrivere codice modulare e mantenibile per rispondere agli input dell'utente in modo efficiente.