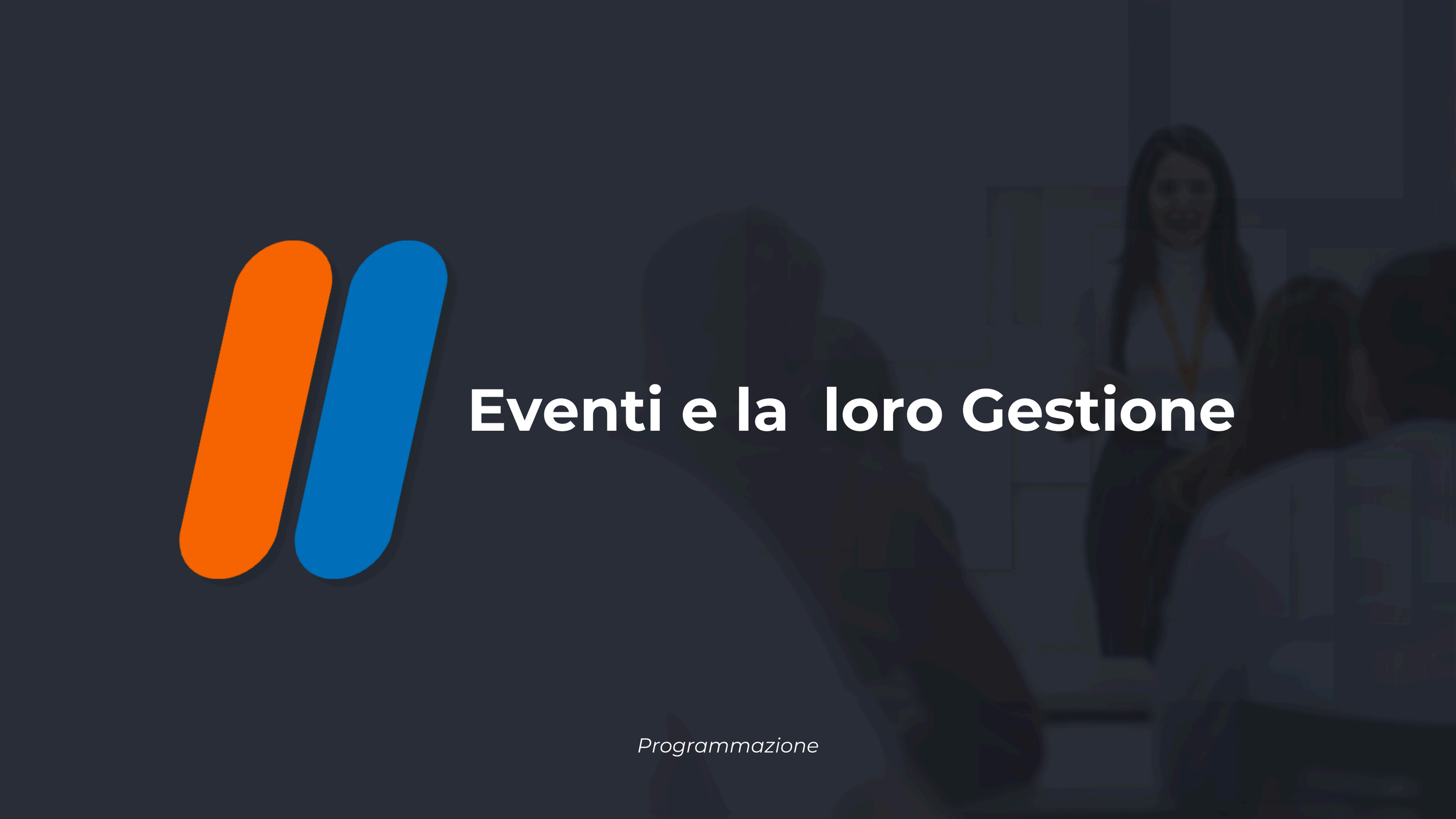


Eventi e la loro Gestione

Programmazione

In JavaScript, gli eventi sono azioni o accadimenti che si verificano nel browser, tipicamente a seguito dell'interazione dell'utente con la pagina web. Esempi comuni includono:

- Il click su un pulsante
- Il passaggio del mouse su un elemento
- La pressione di un tasto
- Il caricamento della pagina
- L'invio di un form

Questi eventi non fanno nulla da soli: per reagire ad essi, bisogna associare dei gestori (event handlers), ovvero funzioni che dicono al browser "cosa fare quando questo evento accade".

La gestione degli eventi consiste nel "ascoltare" (listen) un evento su un elemento del DOM, e poi "rispondere" eseguendo una funzione quando quell’evento si verifica.

Ci sono tre modi principali per gestire eventi in JavaScript:

1. **<button onclick="saluta()">Cliccami</button>**

(poco consigliato perché mescola HTML e JS)

- **Proprietà JavaScript**

```
1. const btn = document.getElementById("miaBtn");
2. btn.onclick = function () {
3.   alert("Hai cliccato!");
4. };
```

- **Metodo addEventListener ✅ (più moderno e flessibile)**

```
1. const btn = document.getElementById("miaBtn");
2. btn.addEventListener("click", function () {
3.   alert("Evento gestito con addEventListener!");
4. });
```

Vantaggi di addEventListener

- Puoi assegnare più gestori allo stesso evento
- Funziona anche con eventi personalizzati
- Permette una migliore separazione tra logica e struttura HTML

Glossario Minimo

- **evento**

Azione che avviene nel browser

- **handler**

Funzione che reagisce all’evento

- **target**

Elemento HTML su cui l’evento è ascoltato

- **event object**

Oggetto passato al gestore con info sull’evento

Eventi in JavaScript: Concetto Base

In JavaScript, gli eventi rappresentano qualsiasi interazione che può avvenire tra l'utente e una pagina web.

Gli eventi non sono limitati ai clic: includono movimenti del mouse, digitazioni sulla tastiera, scroll, caricamento della pagina, invio di un modulo, e molti altri.

Questi eventi sono rilevati dal browser, che consente a JavaScript di rispondere in modo dinamico.

Grazie agli eventi, possiamo rendere le pagine interattive e reattive, migliorando l'esperienza utente.

Gestione degli Eventi: Ascoltare e Reagire

Per rispondere a un evento, JavaScript utilizza i gestori di eventi (event handlers).

Il modo più consigliato e moderno per aggiungere un gestore è tramite addEventListener(), che permette di ascoltare un evento su un elemento del DOM e di associare una funzione di callback che verrà eseguita quando l'evento si verifica.

Questo metodo è preferibile rispetto all'uso diretto degli attributi HTML (onclick, onchange, ecc.) perché separa la logica dal layout e consente una gestione modulare e riutilizzabile del codice

Esempio Completo con Spiegazione

Supponiamo di voler mostrare un messaggio quando un utente clicca un bottone:

```
<!-- HTML -->
<button id="cliccaBtn">Cliccami</button>
<p id="output"></p>

<script>
  // 1. Selezioniamo il bottone e il paragrafo
  const bottone = document.getElementById("cliccaBtn");
  const output = document.getElementById("output");

  // 2. Aggiungiamo un ascoltatore per l'evento 'click'
  bottone.addEventListener("click", function () {
    // 3. Questo codice si attiva quando l'utente clicca
    output.textContent = "Hai cliccato il bottone!";
  });
</script>
```

Spiegazione punto per punto

<button id="cliccaBtn">Cliccami</button>

- **→ Creiamo un bottone con un identificatore unico (id) per poterlo selezionare in JS.**

** **

- **→ Paragrafo vuoto che verrà riempito dinamicamente quando si verifica l'evento.**

document.getElementById("cliccaBtn")

- **→ In JS, selezioniamo il bottone usando il suo id e lo salviamo in una variabile (bottone).**

addEventListener("click", function() { ... })

- **→ Diciamo al browser: "Quando l'utente clicca su questo bottone, esegui questa funzione".**

output.textContent = "Hai cliccato il bottone!";

- **→ Modifichiamo dinamicamente il contenuto del paragrafo, reagendo all'interazione dell'utente.**

Ecco una lista dei principali eventi disponibili in JavaScript, suddivisi per categoria, con una breve descrizione utile per la didattica o per costruire esempi:

🖱️ Eventi del Mouse

- **click**

L'utente fa clic su un elemento

- **dblclick**

Doppio clic sull'elemento

- **mousedown**

Pulsante del mouse premuto

- **mouseup**

Rilascio del pulsante del mouse

- **mousemove**

Movimento del mouse sull'elemento

- **mouseover**

Il cursore entra nell'area dell'elemento

- **mouseout**

Il cursore esce dall’area dell’elemento

- **contextmenu**

Tasto destro del mouse (menu contestuale)

⌨️ Eventi da Tastiera

- **keydown**

Tasto premuto (ripetuto se tenuto premuto)

- **keyup**

Tasto rilasciato

- **keypress**

Tasto premuto (deprecated, non usarlo nei progetti nuovi)

📄 Eventi sul Documento / Pagina

- **load**

La pagina o risorsa è completamente caricata

- **DOMContentLoaded**

Il DOM è stato caricato, ma non le immagini

- **resize**

La finestra del browser cambia dimensione

- **scroll**

L’utente scorre la pagina

📥 Eventi su Form e Input

- submit

Invio di un form

- change

Cambiamento nel valore di un input

- input

Modifica in tempo reale del valore (input, textarea)

- focus

Un elemento riceve il focus (es. input selezionato)

- blur

Un elemento perde il focus

📦 Eventi Drag & Drop

- dragstart

Inizio del trascinamento

- dragover

Elemento trascinato sopra un'area valida

- drop

Elemento rilasciato su un’area

- dragend

Fine dell’operazione di drag

Eventi Personalizzati in JavaScript

In JavaScript non siamo limitati a usare solo gli eventi predefiniti come click, keydown o submit.

Possiamo anche creare eventi personalizzati (custom events), utili quando vogliamo far "parlare" tra loro parti diverse dell'applicazione.

Questa funzionalità è molto potente soprattutto nei progetti modulari o nelle applicazioni complesse, come giochi, interfacce reattive, o componenti riutilizzabili.

Cos’è un Evento Personalizzato?

Un evento personalizzato è un evento definito da noi, che non esiste di default nel browser, ma che possiamo creare, lanciare e ascoltare come qualsiasi altro evento del DOM.

Si basa su tre passaggi principali:

- Creazione dell’evento con new CustomEvent()
- Ascolto dell’evento con addEventListener()
- Dispatch (invio) dell’evento con dispatchEvent()

```
<!-- HTML -->
<button id="btnLancia">Lancia Evento</button>
<p id="output"></p>

<script>
  // 1. Selezione degli elementi
  const bottone = document.getElementById("btnLancia");
  const output = document.getElementById("output");

  // 2. Aggiungo un listener per l'evento personalizzato 'eventoSpeciale'
  document.addEventListener("eventoSpeciale", function (e) {
    output.textContent = "Evento personalizzato ricevuto! Dato: " + e.detail.messaggio;
  });

  // 3. Quando l'utente clicca, creo e lancio l'evento personalizzato
  bottone.addEventListener("click", function () {
    const evento = new CustomEvent("eventoSpeciale", {
      detail: {
        messaggio: "Ciao dal tuo evento!"
      }
    });
    document.dispatchEvent(evento);
  });
</script>

```

- **Spiegazione punto per punto**

document.addEventListener("eventoSpeciale", …)

- **→ Si ascolta un evento chiamato "eventoSpeciale" anche se non è nativo. Quando verrà inviato, si eseguirà la funzione.**

new CustomEvent("eventoSpeciale", { detail: { … } })

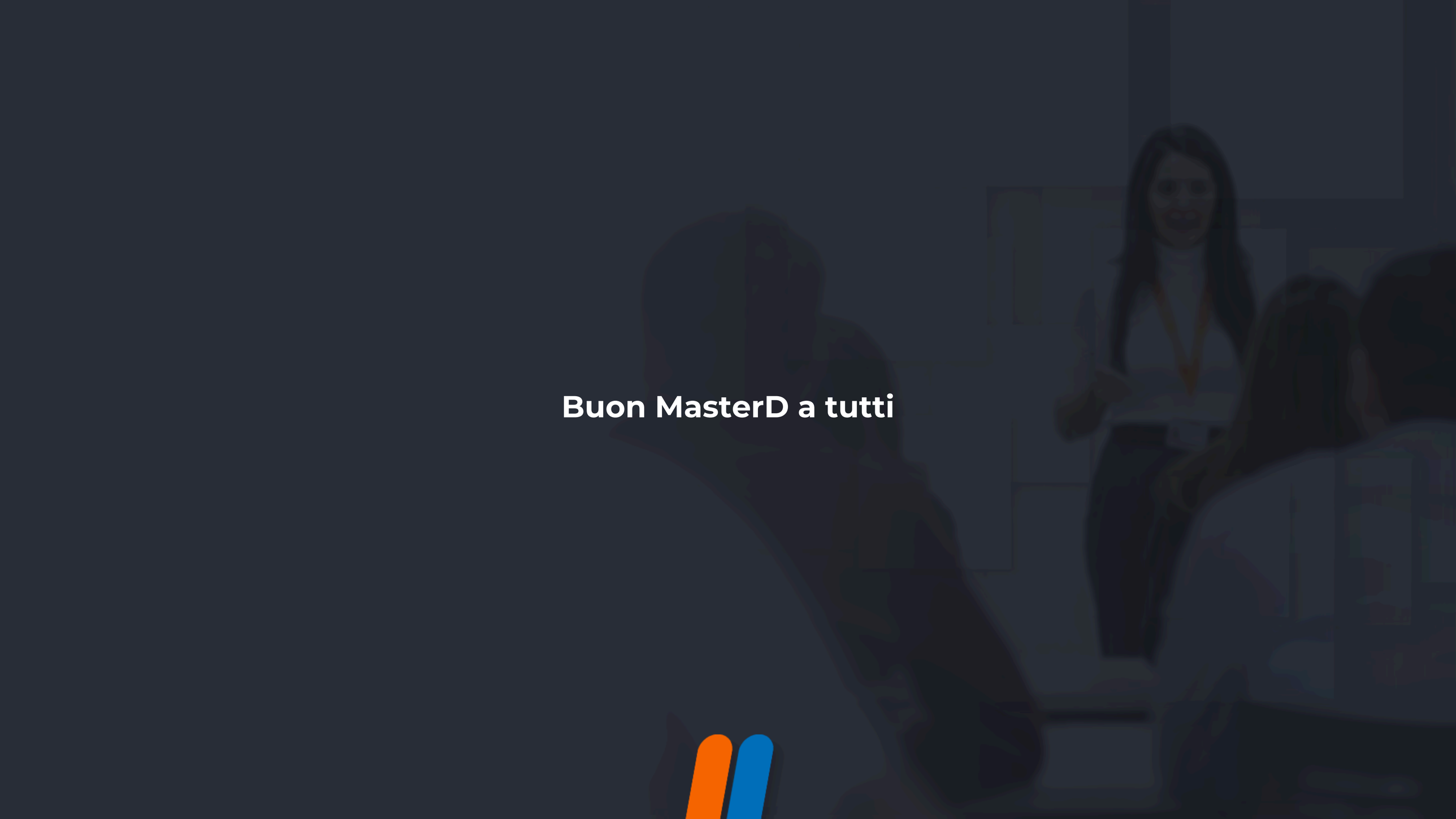
- **→ Si crea un evento personalizzato. Il secondo parametro (detail) serve a passare dati aggiuntivi all'evento.**

dispatchEvent(evento)

- **→ Il metodo che "lancia" l'evento, ovvero lo propaga nel DOM dove può essere intercettato da chi lo ascolta.**

e.detail.messaggio

- **→ All'interno del gestore, e è l'oggetto evento. La proprietà detail contiene i dati che abbiamo passato all'evento.**


Buon MasterD a tutti