Gestire gli Eventi

- Il DOM consente di reagire alle **azioni dell'utente** (click, mouseover, tasti premuti, ecc.).
- Si utilizzano gli **event listener** per "ascoltare" specifici eventi su specifici elementi e eseguire codice in risposta.
- `elemento.addEventListener('tipoEvento', funzioneCallback)`: Attacca un gestore di eventi all'elemento.

```
let pulsanteInterattivo = document.getElementById('interattivo');
pulsanteInterattivo.addEventListener('click', function() {
  alert('Hai cliccato il pulsante!');
});
```

Cos'è un Evento?

Un **evento** è un'azione o un'occorrenza che succede nel browser (o all'interno della pagina web).

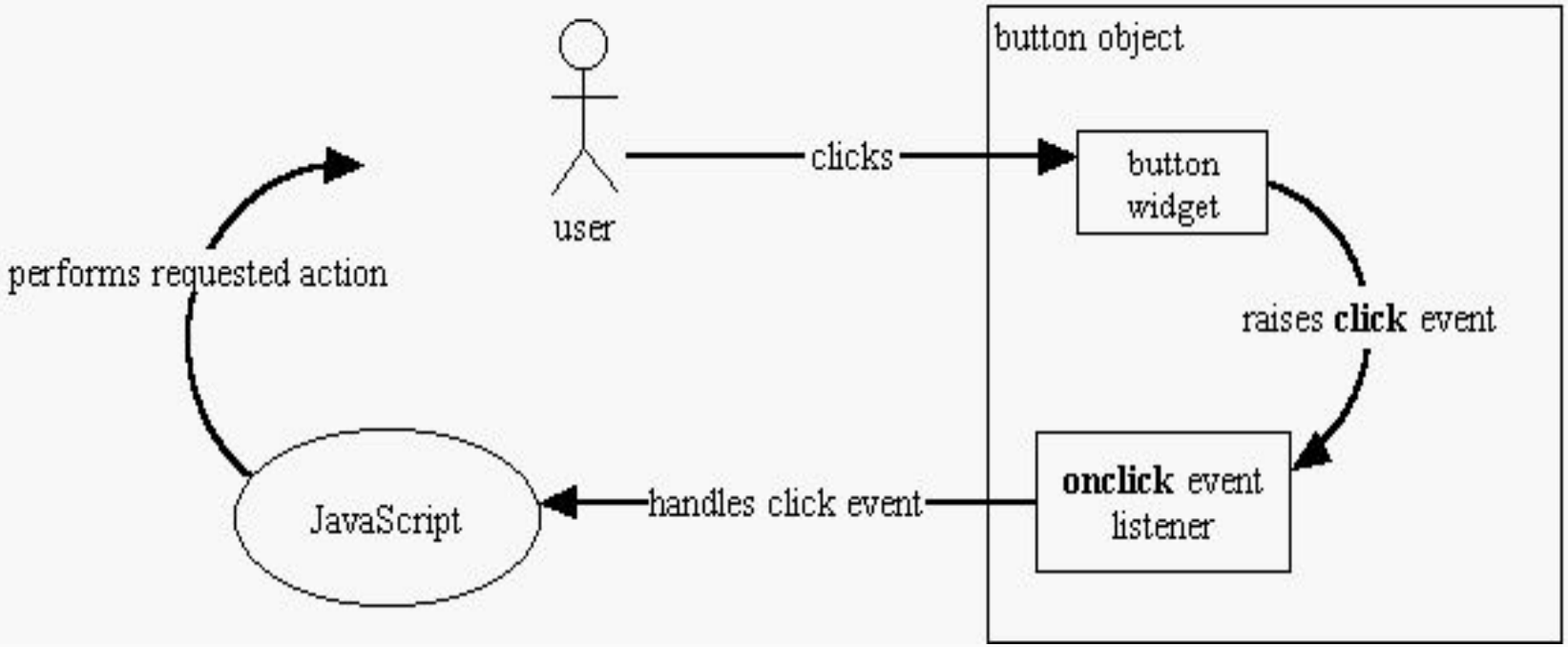
Può essere causato dall'**utente** (es. clic del mouse, pressione di un tasto, movimento del mouse) o dal **browser** stesso (es. caricamento della pagina, ridimensionamento della finestra, errore).

https://www.w3schools.com/js/js_events.asp

Esempi di Eventi Comuni:

- `click` : Clic su un elemento.
- `mouseover` : Il puntatore del mouse si sposta sopra un elemento.
- `mouseout` : Il puntatore del mouse esce da un elemento.
- `keydown` : Un tasto viene premuto.
- `keyup` : Un tasto viene rilasciato.
- `submit` : Un form viene inviato.
- `change` : Il valore di un elemento di input cambia.
- `load` : La pagina o una risorsa (es. immagine) è stata caricata completamente.
- `resize` : La finestra del browser è stata ridimensionata.
- `scroll` : La barra di scorrimento è stata spostata.

```
<button value="Click Me" onclick="alert('Thank you')" />
```



I Listener di Eventi (Event Listeners)

- Un **event listener** è una funzione che "ascolta" un particolare evento su uno specifico elemento HTML.
- Quando l'evento si verifica, la funzione listener (chiamata anche **callback dell'evento** o **gestore dell'evento**) viene eseguita.

`tipoEvento`: Una stringa che specifica il tipo di evento da ascoltare (es. `'click'`, `'mouseover'`, `'keydown'`).
`funzioneCallback`: La funzione JavaScript da eseguire quando l'evento si verifica.

```
let mioPulsante = document.getElementById('bottone');

function gestisciClick() {
  console.log('Il pulsante è stato cliccato!');
}

mioPulsante.addEventListener('click', gestisciClick);
```

I Listener di Eventi (Event Listeners)

- Quando un evento si verifica e il suo listener viene eseguito, una speciale **oggetto evento** viene automaticamente passato alla funzione callback come primo argomento.
- Questo oggetto contiene informazioni dettagliate sull'evento che si è verificato.

Proprietà Utili dell'Oggetto Evento:

- `type`: Il tipo di evento (es. `'click'`, `'mouseover'`).
- `target`: L'elemento HTML su cui l'evento si è verificato (l'elemento "bersaglio").
- `currentTarget`: L'elemento HTML a cui l'event listener è stato effettivamente collegato (potrebbe essere diverso da `target` in caso di event delegation).
- `clientX`, `clientY`: Le coordinate del puntatore del mouse rispetto alla finestra del browser.
- `offsetX`, `offsetY`: Le coordinate del puntatore del mouse rispetto all'elemento target.
- `key`: Il tasto premuto (per eventi `keydown` e `keyup`).
- `keyCode` (deprecato, usare `key`): Il codice numerico del tasto premuto.
- `preventDefault()`: Un metodo che impedisce il comportamento predefinito dell'evento (es. impedire l'invio di un form).
- `stopPropagation()`: Un metodo che impedisce la propagazione dell'evento attraverso il DOM (event bubbling e capturing).

I Listener di Eventi (Event Listeners)

- È importante rimuovere i listener di eventi quando non sono più necessari per evitare memory leak e comportamenti inattesi.
- Si utilizza il metodo `removeEventListener()`, passando gli stessi argomenti usati per aggiungerlo.

Per rimuovere un listener, la funzione callback passata a `removeEventListener` deve essere **la stessa identica funzione** (stessa referenza in memoria) passata a `addEventListener`. Se si usa una funzione anonima direttamente in `addEventListener`, non sarà possibile rimuoverla in seguito (a meno di non conservare un riferimento alla funzione).

```
let altroPulsante = document.getElementById('bottoneUnico');

function gestisciClickUnico() {
  console.log('Questo messaggio apparirà solo una volta.');
  altroPulsante.removeEventListener('click', gestisciClickUnico);
}

altroPulsante.addEventListener('click', gestisciClickUnico);
```

Event Bubbling e Event Capturing

Event Bubbling (Fase di "bolla"): Quando un evento si verifica su un elemento, si propaga ("risale") l'albero del DOM, attivando gli event listener degli elementi genitori. Questo è il comportamento predefinito.

Event Capturing (Fase di "cattura"): È una fase opzionale in cui l'evento si propaga "verso il basso" attraverso l'albero del DOM, attivando gli event listener degli elementi antenati **prima** di raggiungere l'elemento target.

Il terzo argomento opzionale di `addEventListener()` specifica se il listener deve essere registrato per la fase di capturing (`true`) o di bubbling (`false` o omesso).

Usa `preventDefault()` e `stopPropagation()` con cautela: Potrebbero interrompere comportamenti predefiniti importanti.

```
document.getElementById('padre').addEventListener('click', function() {
  console.log('Click sul padre (bubbling)');
}, false);

document.getElementById('figlio').addEventListener('click', function() {
  console.log('Click sul figlio (bubbling)');
}, false);

document.getElementById('nonno').addEventListener('click', function() {
  console.log('Click sul nonno (capturing)');
}, true);

document.getElementById('nipote').addEventListener('click', function() {
  console.log('Click sul nipote (capturing)');
}, true);
```