

# Gli eventi in Javascript

di Roberta Molinari

# JavaScript

## Gli eventi

---

Per gestire un evento si deve invocare il suo gestore (handler):

1. Assegnando l'handler (possono anche essere istruzioni) direttamente all'attributo associato all'evento nel tag HTML:

```
<IMG onClick="nomeFunz () ">
```

2. Associando la funzione di callback alle relative proprietà dell'oggetto:

```
window.onResize=nomeFunz; //senza parentesi
```

- 4 Tramite `addEventListener` sull'oggetto DOM (così si ha maggiore flessibilità)

```
body.addEventListener("onLoad", nomeFunz)
```

# JavaScript

## Gli eventi

---

Se si utilizza il primo metodo e la funzione restituisce true, viene anche eseguita l'operazione associata di default all'evento, se restituisce false non viene eseguita:

```
<a href="#" onClick="nomeFunz()">
```

"#" indica di posizionarsi ad inizio pagina, se  
nomeFunz() restituisce false, non verrà eseguita

```
<input type="submit"  
onClick="return confirm('Confermi?')>
```

# JavaScript

## Gli eventi

---

**addEventListener** ("evento", funz)

Si può usare in questo modo

```
document.addEventListener("click", myFunction);  
function myFunction() {  
    document.getElementById("demo").innerHTML =  
    "Hello World";}
```

o in questo

```
document.addEventListener("click", function() {  
    document.getElementById("demo").innerHTML =  
    "Hello World";});
```

- ▶ Se l'oggetto ha già un gestore per quell'evento, non si sovrascrive, ma si aggiunge
- ▶ Per eliminare un handler

**.removeEventListener** ("click", myFunction)

---













# JavaScript

## Gli eventi

Evento	Tag	Descrizione
abort	<img>	L'utente fallisce il caricamento di un'immagine.
blur	<body> <frameset> <frame> <input type = "text"> <textarea> <select>	Un documento perde il focus dell'input. Un frame perde il focus dell'input. Un frame perde il focus dell'input. Un campo di testo perde il focus dell'input. Un'area di testo perde il focus dell'input. Un elemento di selezione perde il focus dell'input.
change	<input type = "text"> <textarea> <select>	Un campo di testo viene modificato e perde il focus dell'input. Un'area di testo viene modificata e perde il focus dell'input. Un elemento di selezione viene modificato e perde il focus dell'input.
click	<a> <input type = "button"> <input type = "submit"> <input type = "reset"> <input type = "radio"> <input type = "checkbox">	L'utente fa clic su un collegamento. Viene selezionato un pulsante. Viene selezionato il pulsante Submit. Viene selezionato il pulsante Reset. Viene selezionato un pulsante di opzione. Viene selezionata una casella di controllo.

# JavaScript

## Gli eventi

Evento	Tag	Descrizione
error	  	<p>Si è verificato un errore nel caricamento di un'immagine.</p> <p>Si è verificato un errore nel caricamento di un documento.</p> <p>Si è verificato un errore nel caricamento di un set di frame.</p>
focus	     	<p>Un documento diventa attivo con l'input.</p> <p>Un set di frame diventa attivo con l'input.</p> <p>Un frame viene diventa attivo con l'input.</p> <p>Un campo di testo diventa attivo con l'input.</p> <p>Un'area di testo diventa attiva con l'input.</p> <p>Un elemento di selezione diventa attivo con l'input.</p>
load	  	<p>L'utente ha caricato e visualizzato un'immagine.</p> <p>Il caricamento del documento è completato.</p> <p>Il caricamento del documento è completato.</p>

# JavaScript

## Gli eventi

Evento	Tag	Descrizione
mousemove		L'utente muove il mouse
mouseout	<a> <area>	L'utente allontana il cursore del mouse dal collegamento. Il cursore del mouse viene spostato fuori dalla mappa immagine.
mouseover	<a> < area>	Il cursore del mouse viene spostato su un collegamento. Il cursore del mouse viene spostato sull'area di una mappa immagine.
reset	<form>	Viene selezionato il pulsante Reset.
resize	<body> <frameset>	L'utente modifica le dimensioni della finestra. L'utente modifica le dimensioni del frame
select	<input type "text"> <textarea>	Il cursore del mouse viene spostato su un campo di testo. Il cursore del mouse viene spostato all'interno di un'area di testo.
submit	<form>	Viene selezionato il pulsante Submit.
unload	<body> <frameset>	L'utente esce dal documento. L'utente esce dal set di frame.

- Perchè il body possa sentire degli eventi del mouse, deve avere un testo o un'altezza impostata (`style="height:500px"`)

# JavaScript

## L'oggetto Event: propagazione

Secondo le specifiche del W3C, la propagazione di un evento avviene in tre fasi:

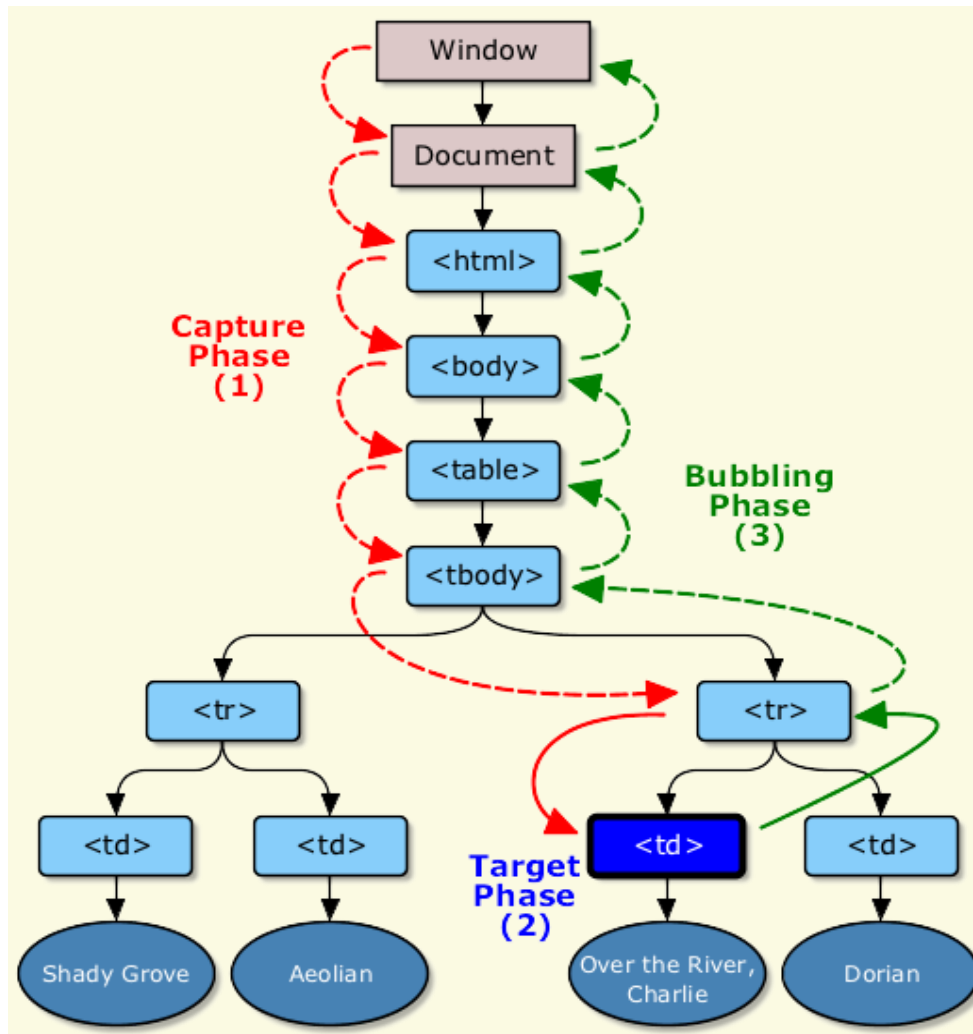
<b>Capture phase</b>	In questa fase l'evento si propaga dalla radice del DOM verso l'elemento destinatario effettivo
<b>Target phase</b>	In questa fase l'evento raggiunge l'elemento destinatario
<b>Bubble phase</b>	Questa è la fase in cui l'evento risale l'albero del DOM partendo dall'elemento target fino a raggiungere la radice, passando quindi dagli stessi nodi attraversati nella fase di cattura.

Durante le 3 fasi, un oggetto *event* associato all'evento viene passato agli eventuali gestori incontrati lungo il cammino



# JavaScript

## L'oggetto Event: propagazione



► Normalmente, nella fase di **capturing** i gestori ignorano l'evento e lo fanno fluire verso l'elemento target.

► Una volta raggiunto l'elemento **target** viene eseguito il codice del gestore associato all'evento.

► Nella fase di **bubbling** vengono eseguiti i gestori dell'evento che si incontrano man mano che si va verso la radice del DOM

# JavaScript

## L'oggetto Event

- Il DOM prevede che ad ogni gestore di eventi venga passato come parametro l'oggetto **event** contenente informazioni su di esso.

```
function gestoreEvento(e) {  
    //e contiene un'istanza dell'oggetto event  
    e.target.id  
    //id dell'oggetto che ha scatenato l'evento  
}
```

- Per assegnare un event handler attraverso l'HTML che gestisca l'oggetto *event*, occorre specificarlo tra i parametri passati

```
<p id="myP" onclick="gestoreEvento(event)">...
```

mentre non cambia nulla se l'evento è definito via Javascript

```
document.getElementById("myP").onclick =  
    gestoreEvento
```

# JavaScript

## L'oggetto Event: Proprietà

---

- ▶ `target` restituisce un riferimento al nodo obiettivo del flusso d'evento nella sua fase di capturing (punto di partenza della successiva fase di bubbling)
- ▶ `relatedTarget` nel caso di un evento di *onmouseover*, la proprietà contiene un riferimento all'elemento dal quale il mouse proviene, cioè l'elemento appena lasciato. Nel caso dell'*onmouseout*, contiene un riferimento all'elemento verso il quale il mouse è diretto, ovvero l'elemento nel quale ci si sposta.
- ▶ `currentTarget` restituisce un riferimento al nodo per il quale il flusso d'evento sta passando

# JavaScript

## L'oggetto Event: Proprietà

---

- ▶ `screenX`, `screenY` restituisce la coordinata X/Y del puntatore del mouse relativa allo schermo
  - ▶ `x`, `y` restituisce la coordinata X/Y rispetto alla finestra del browser
  - ▶ `button` restituisce quale pulsante del mouse ha cambiato il proprio stato, ovvero sia stato premuto o rilasciato. I valori restituiti possono essere:
    - 0 : pulsante sinistro [1 per Netscape]
    - 1 : pulsante centrale se possiede 3 pulsanti [2 per Netscape]
    - 2 : pulsante destro [3 per Netscape]
  - ▶ `charCode` codice del carattere Unicode premuto da tastiera
-

# JavaScript

## L'oggetto Event: Proprietà

---

- ▶ `type` restituisce una stringa che descrive il tipo di evento, come ad esempio: "click", "mouseover", ecc
- ▶ `cancelable` indica se l'azione di default di un evento possa essere cancellata (true|false)
- ▶ `preventDefault()` consente di cancellare l'azione di default per quegli eventi per cui la proprietà `cancelable` restituisce il valore true
- ▶ `stopPropagation` questo metodo permette di interrompere la propagazione dell'evento, arrestando l'*Event Flow* indipendentemente che sia nella sua fase di capturing o in quella di bubbling

# JavaScript

## L'oggetto Event: esempio

---

```
<div id="mainDiv">  
    <p id="p1">Clicca su questo paragrafo</p>  
    <p id="p2">Altro paragrafo</p>  
</div>
```

...

```
var myDiv = document.getElementById("mainDiv");  
var myP = document.getElementById("p1");  
var handler = function() { alert(this.id) };  
myDiv.addEventListener("click", handler);  
myP.addEventListener("click", handler);
```

Otteniamo prima l'id del primo paragrafo e poi quello del <div>.

# JavaScript

## L'oggetto Event: esempio

---

- Per invertire l'ordine di gestione si usa un terzo parametro opzionale del metodo `addEventListener()` che abilita la gestione dell'evento nella fase di capturing

```
myDiv.addEventListener("click", handler, true);
```

- Per fare in modo che venga eseguito solo un gestore dell'evento si blocca la propagazione

```
var handler = function(e) {  
    console.log(this.id);  
    //this elemento su cui si è verificato l'evento  
    e.stopPropagation();  
};
```

---

# JavaScript

## Drag & drop: eventi

Evento	Tag	Descrizione
ondragstart	tutti gli elementi con attributo draggable="true"	si sta trascinando l'elemento
ondragover	tutti gli elementi con attributo draggable="true"	quando un elemento viene trascinato su una destinazione di rilascio valida
ondrop	tutti gli elementi con attributo draggable="true"	si è rilasciato un elemento sopra questo

- `.dataTransfer` Restituisce un oggetto contenente i dati trascinati/rilasciati o inseriti/eliminati. Il metodo `dataTransfer.setData("key",value)` imposta una coppia chiave/valore e `getData("key")` restituisce il valore.
- Per impostazione predefinita, gli elementi trascinati non possono essere rilasciati in altri elementi. Per consentire un inserimento, dobbiamo impedire la gestione predefinita dell'elemento trascinato e dell'elemento "ricevente". Si deve chiamare per entrambi il metodo `e.preventDefault()`



# JavaScript

## Drag & drop: esempio

---

Trascina la scritta nell'altro paragrafo

```
<script>
function allowDrop(e) {e.preventDefault();}
function drag(e) {
    e.dataTransfer.setData("text", e.target.id);}
function drop(e) {
    e.preventDefault();
    var data = e.dataTransfer.getData("text");
    e.target.appendChild(document.getElementById(data));
}
</script>

<div id="div1" ondrop="drop(event)"
    ondragover="allowDrop(event)"></div>
<div id="drag1" draggable="true"
    ondragstart="drag(event)">ciao</div>
```

---