

Laboratorio di “Sistemi Distribuiti”

A.A. 2024-2025



JavaScript (DOM)

Emanuele Petriglia

Slide aggiornate al 29 Maggio 2025

Indice dei contenuti

1. Introduzione a DOM

Ambiente di sviluppo

Document Object Model

Esercizio A

2. Selezione degli elementi

Esercizio B

3. Gestione degli eventi

Esercizio C

1. Introduzione a DOM

Materiale utile per JavaScript

- Negli ultimi due laboratori useremo JavaScript per interagire con le pagine Web.
- Dove imparare JavaScript?
 - [Learn JavaScript in Y minutes](#) → introduzione
 - Argomento 7 su e-Learning → introduzione
 - [JavaScript \(MDN\)](#) → tutorial e reference



JavaScript e il Web

- Tecnologie principali del Web:
 - **HTML**: definisce la struttura e il contenuto della pagina,
 - **CSS**: definisce lo stile grafico della pagina (non lo vedremo),
 - **JavaScript**: permette di rendere interattiva e dinamica la pagina.
- Il codice **JavaScript** è **eseguito dal client** dal browser Web (Firefox, Chrome...).
- Il codice JavaScript interagisce con il browser (e quindi con la pagina) tramite delle API (**Web API**).
- **Esistono** decine di Web API: **DOM API**, **Fetch API**, URL API, Canvas API, Fullscreen API, Gamepad API, XMLHttpRequest API...

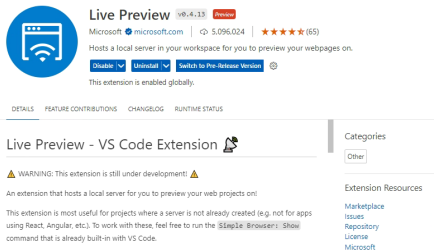


1. Introduzione a DOM

Ambiente di sviluppo

L'estensione "Live Preview" (pt. 1)

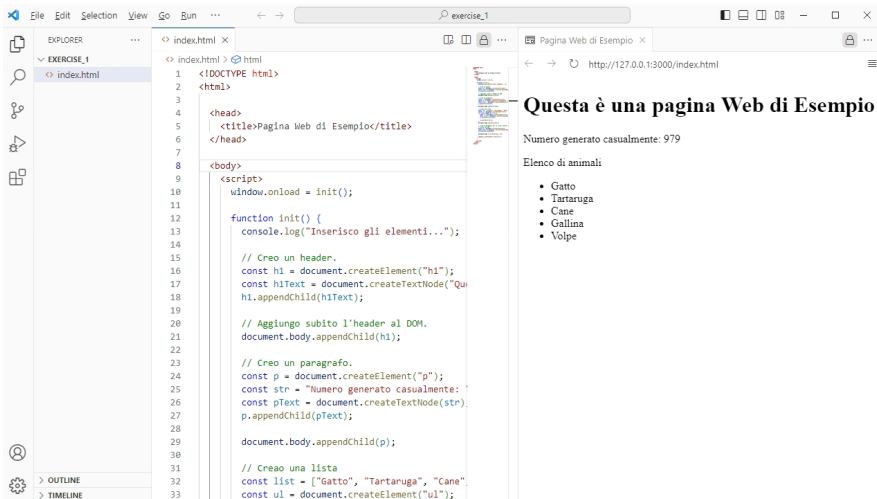
- Verrà usato Visual Studio Code. È necessario installare l'estensione **Live Preview** di Microsoft.



- Live Preview permette di avviare al volo un server Web che espone le pagine HTML locali.
- Scaricare, estrarre ed aprire con VS Code il progetto hello-world. Fare click destro su `index.html` e selezionare Show Preview.



L'estensione "Live Preview" (pt. 2)



The screenshot shows the VS Code interface with the Live Preview extension. The Explorer on the left shows a project named 'EXERCISE_1' with a file 'index.html'. The editor displays the code for 'index.html', which includes HTML structure and JavaScript for DOM manipulation. The Live Preview window on the right shows the rendered page with the title 'Pagina Web di Esempio' and a list of animals.

```
1 <!DOCTYPE html>
2 <html>
3
4   <head>
5     <title>Pagina Web di Esempio</title>
6   </head>
7
8   <body>
9     <script>
10       window.onload = init();
11
12       function init() {
13         console.log("Inserisco gli elementi...");
14
15         // Creo un header.
16         const h1 = document.createElement("h1");
17         const h1Text = document.createTextNode("Qualche esempio");
18         h1.appendChild(h1Text);
19
20         // Aggiungo subito l'header al DOM.
21         document.body.appendChild(h1);
22
23         // Creo un paragrafo.
24         const p = document.createElement("p");
25         const str = "Numero generato casualmente: 979";
26         const pText = document.createTextNode(str);
27         p.appendChild(pText);
28
29         document.body.appendChild(p);
30
31         // Creo una lista
32         const list = ["Gatto", "Tartaruga", "Cane"];
33         const ul = document.createElement("ul");
```

Questa è una pagina Web di Esempio

Numero generato casualmente: 979

Elenco di animali

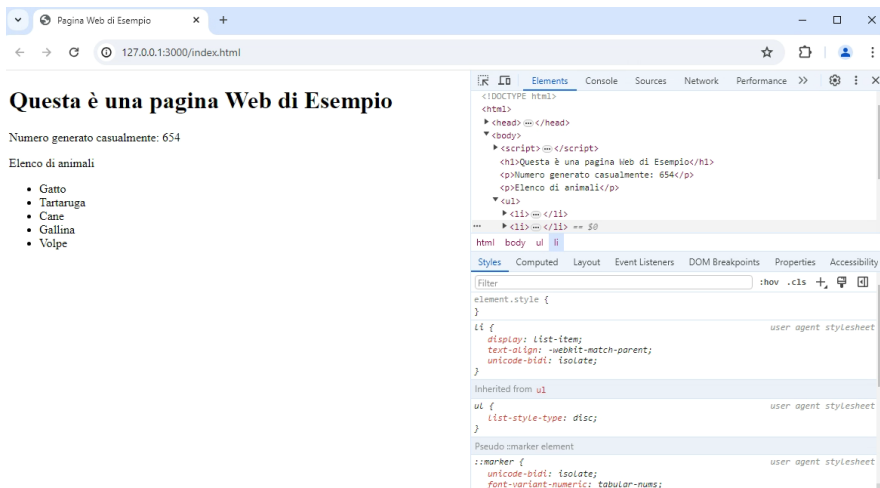
- Gatto
- Tartaruga
- Cane
- Gallina
- Volpe



L'estensione "Live Preview" (pt. 3)

- La pagina Web viene ricaricata appena si modifica il file HTML (*Live Refreshing*).
- Premendo il burger menu a sinistra della barra dell'URL:
 - ① Open in Browser → apre la pagina in Chrome (più comodo).
 - ② Open Devtools pane → apre gli strumenti di sviluppo.
- Selezionare Open in Browser e poi, nel browser, click destro e Ispeziona.

Gli strumenti di sviluppo di Chrome (pt. 1)



The screenshot shows a web browser window with a single tab titled "Pagina Web di Esempio". The address bar shows the URL "127.0.0.1:3000/index.html". The page content includes a heading "Questa è una pagina Web di Esempio", a paragraph "Numero generato casualmente: 654", and a section "Elenco di animali" with a bulleted list of animals: Gatto, Tartaruga, Cane, Gallina, and Volpe.

The Chrome DevTools interface is open on the right side of the browser window. The "Elements" panel is selected, showing the DOM tree. The tree structure is as follows:

- <!DOCTYPE html>
 - <html>
 - <head>
 - <script>
 - <body>
 - <h1>Questa è una pagina Web di Esempio</h1>
 - <p>Numero generato casualmente: 654</p>
 - <p>Elenco di animali</p>
 -
 -
 -

The "Styles" panel is also open, showing the default user agent styles for the selected element (a list item, `li`). The styles include:

- `display: list-item;`
- `text-align: -webkit-match-parent;`
- `unicode-bidi: isolate;`

The "Inherited from" section shows the styles for the `ul` element:

- `list-style-type: disc;`

The "Pseudo::marker element" section shows the default styles for the `::marker` pseudo-element:

- `unicode-bidi: isolate;`
- `font-variant-numeric: tabular-nums;`



Gli strumenti di sviluppo di Chrome (pt. 2)

Alcune schede importanti:

- **Elements:** DOM come albero gerarchico.
- **Console:** mostra la console (interattiva) dell'ambiente JavaScript.
- **Sources:** file originali che provengono dal server.
- **Network:** cronologia delle chiamate HTTP.



1. Introduzione a DOM

Document Object Model

Cos'è il DOM?

- **DOM** (Document Object Model) è una rappresentazione in memoria della struttura di una pagina Web.
- Strutturato come albero.
- Il browser renderizza graficamente la pagina Web leggendo il DOM.
- La struttura è manipolabile! E ogni modifica viene subito renderizzata dal browser (se la modifica è visiva).



Pagina di esempio...



Questa è una pagina Web di Esempio

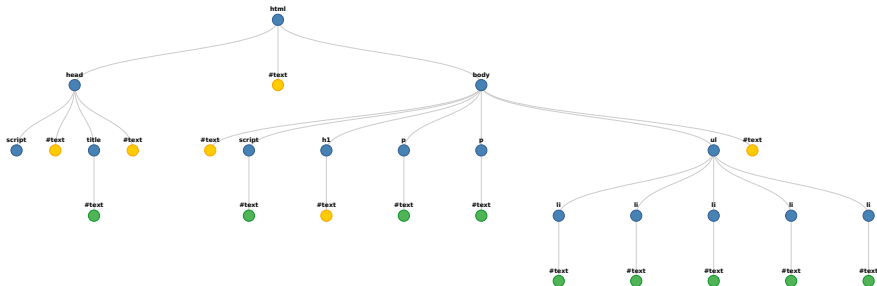
Numero generato casualmente: 654

Elenco di animali

- Gatto
- Tartaruga
- Cane
- Gallina
- Volpe



...e DOM corrispondente



Accedere al DOM (pt. 1)

- Due variabili globali esposte per JavaScript:
 - `window`: rappresenta una finestra del browser, con tante proprietà e funzioni, tra cui `document` che è il DOM.
 - `document` rappresenta il DOM, con varie proprietà e funzioni, tra cui l'accesso agli elementi dell'albero (è una scorciatoia a `window.document`).
- Dove scrivere codice JavaScript? All'interno dei tag `<script>`.
 - In fondo al `<body>`.
 - All'interno di `<script>` si possono richiamare script esterni. Nei laboratori scriveremo JavaScript nella stessa pagina.

Attenzione

Non mischiare codice HTML con JavaScript. Il codice JavaScript va sempre all'interno del tag `<script>`.



Accedere al DOM (pt. 2)

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>Pagina Web di Esempio</title>
5    </head>
6    <body>
7      <script>
8        window.onload = init();
9
10       function init() {
11         console.log("Hello World!");
12
13         const paragraph = document.createElement("p");
14         const text = document.createTextNode("Hello World!");
15         paragraph.appendChild(text);
16         document.body.appendChild(paragraph);
17       }
18     </script>
19   </body>
20 </html>
```



Creazione di una lista (pt. 1)

```
1  function init() {
2      const colors = ["Rosso", "Giallo", "Blu", "Verde", "Nero",
3          ↪  "Bianco", "Arancione"];
4      const list = document.createElement("ul");
5      for (let color of colors) {
6          console.log("Aggiunta dell'elemento ", color);
7
8          const item = document.createElement("li");
9          item.appendChild(document.createTextNode(color));
10         list.appendChild(item);
11     }
12     document.body.appendChild(list);
13 }
```



Creazione di una lista (pt. 2)

Lista di colori

127.0.0.1:3000/helloworld_list.html

Lista di colori

- Rosso
- Giallo
- Blu
- Verde
- Nero
- Bianco
- Arancione

Elements

```
</head>
<body>
  <h1>Lista di colori</h1>
  <script>
    window.onload = init();

    function init() {
      const colors = ["Rosso", "Giallo", "Blu", "Verde", "Nero", "Bianco",
        "Arancione"];
      const list = document.createElement("ul");
      for (let color of colors) {
        console.log("Aggiunta dell'elemento ", color);

        const item = document.createElement("li");
        item.appendChild(document.createTextNode(color));
        list.appendChild(item);
      }

      document.body.appendChild(list);
    }
  </script>
  <ul>
    <li>
      ::marker
      "Rosso"
    </li>
    <li>
      ::marker
      "Giallo"
    </li>
```



Creazione di una tabella (pt. 1)

```
1  class Lecture {
2      constructor(date, subject, start, end, place) {
3          this.date = date;
4          this.subject = subject;
5          this.start = start;
6          this.end = end;
7          this.place = place;
8      }
9  }
10
11  const headers = ["Data", "Contenuto", "Ora inizio", "Ora fine", "Luogo"];
12
13  const lectures = [
14      new Lecture("2025-05-10", "Servlet", "9:30", "12:30", "Laboratorio A1
15      ↪ (U14)"),
16      new Lecture("2025-05-17", "REST", "9:30", "12:30", "Laboratorio A1
17      ↪ (U14)"),
18      new Lecture("2025-05-24", "DOM", "9:30", "12:30", "Laboratorio A1
19      ↪ (U14)"),
20      new Lecture("2025-05-29", "AJAX", "14:30", "17:30", "Laboratorio 05
21      ↪ (U9)"),
22  ]
```



Creazione di una tabella (pt. 2)

```
1  const table = document.createElement("table");
2  const caption = table.createCaption();
3  caption.appendChild(document.createTextNode("Lezioni Turno 1"));
4  const thead = document.createElement("thead");
5  const theadRow = thead.insertRow();
6  for (let header of headers) {
7    const cell = document.createElement("th");
8    cell.appendChild(document.createTextNode(header));
9    theadRow.appendChild(cell);
10 }
11 table.tHead = thead;
12 const body = table.createTBody();
13 for (let lecture of lectures) {
14   const row = body.insertRow();
15   for (let property in lecture) {
16     row.insertCell().textContent = lecture[property];
17   }
18 }
19 document.body.appendChild(table);
```



Creazione di una tabella (pt. 3)

Tabella delle lezioni

127.0.0.1:3000/helloworld_table.html

Tabella delle lezioni

Lezioni Turno 1				
Data	Contenuto	Ora inizio	Ora fine	Luogo
2024-05-10	Servlet	9:30	12:30	Laboratorio A1 (U14)
2024-05-17	REST	9:30	12:30	Laboratorio A1 (U14)
2024-05-24	DOM	9:30	12:30	Laboratorio A1 (U14)
2024-05-29	AJAX	14:30	17:30	Laboratorio 05 (U9)

Elements

Console

Sources

```
<!DOCTYPE html>
<!-- Pagina di esempio che inserisce al c
tabella. -->
<html>
  <head>
  </head>
  <body>
    <h1>Tabella delle lezioni</h1>
    <script>
    </script>
    <table>
      <caption>Lezioni Turno 1</caption>
      <thead>
        <tr>
          <th>Data</th>
          <th>Contenuto</th>
          <th>Ora inizio</th>
          <th>Ora fine</th>
          <th>Luogo</th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <td>2024-05-10</td>
          <td>Servlet</td>
          <td>9:30</td>
          <td>12:30</td>
          <td>Laboratorio A1 (U14)</td>
        </tr>
      </tbody>
    </table>
  </body>
</html>
```



Interfacce fondamentali

Nell'esempio della tabella è stata usata l'interfaccia `HTMLTableElement` per aggiungere le righe e la didascalia.

Quali altre interfacce esistono che è importante conoscere?

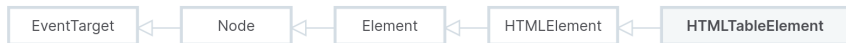
- `Document`: interfaccia che rappresenta la pagina Web, con metodi e proprietà importanti come `createElement()`, `body` e `querySelectorAll()`.
- `EventTarget`: interfaccia implementata da oggetti che reagiscono a eventi con il metodo `addEventListener()`.
- `Node`: interfaccia astratta implementata da tutti i nodi DOM con metodi utili.
- `Element`: interfaccia base per tutti gli oggetti che rappresentano un elemento.
- `HTMLElement`: interfaccia specifica per gli oggetti che rappresentano un elemento HTML (es. un paragrafo `<p>`).



Interfacce multiple

Attenzione!

Un oggetto può implementare più interfacce. Per esempio l'oggetto che rappresenta una tabella HTML (<table>) implementa le seguenti interfacce:



Si può invocare o accedere qualsiasi metodo o proprietà di ogni interfaccia che l'oggetto implementa!

1. Introduzione a DOM

Esercizio A

Consegna (pt. 1)

- Scrivere una pagina HTML che, al suo caricamento, mostri una tabella con le informazioni sulle tastiere.
- La tabella deve essere generata con JavaScript e i dati presi dalla variabile globale `json` presente nello scheletro.
- La tabella deve avere un bordo e una didascalia posta in alto.
- **Partire dallo scheletro** su e-Learning.

Attenzione

Fare l'esercizio **prima** senza usare l'interfaccia `HTMLTableElement`, **dopo** ripetere l'esercizio usando tale interfaccia.



Consegna (pt. 2)

Il risultato deve essere uguale al seguente:



Lista delle tastiere

name	manufacturer	year	ergonomic	backlight	layout
K55 RGB PRO	Corsair	2023	true	true	QUERTY
K30	Corsair	2020	false	false	QUERTY
G213	Logitech	2020	false	true	Dvorak

Suggerimenti

- Come aggiungere il bordo alla tabella o posizionare la didascalia?
L'elemento `table` e `caption` supportano degli attributi¹ utili per tale scopo. Un attributo può essere impostato con il metodo `setAttribute()`:

```
1  const caption = document.createElement("caption");  
2  caption.setAttribute("id", "idDidascalia");
```
- Per fare deserializzare JSON si può usare il metodo statico `JSON.parse()`. Per serializzare: `JSON.stringify()`.

¹Spesso sono attributi deprecati perché sostituiti da CSS, ma nel laboratorio e nel progetto va bene usarli perché non viene introdotto CSS.

2. Selezione degli elementi

Come selezionare un elemento nel DOM?

- ① Tramite metodi `getElementByXXX()` di `Document` che restituiscono una lista di elementi o un solo elemento in base a dei criteri,
- ② Tramite i selettori CSS con i metodi `querySelector()` e `querySelectorAll()` di `Document`.

Metodi getElementByXXX

- `getElementById()`: restituisce l'elemento con l'ID fornito (univoco all'interno della pagina).
- `getElementsByClassName()`: restituisce gli elementi che sono della classe fornita come argomento.
- `getElementsByName()`: restituisce gli elementi che hanno l'attributo name uguale a quello fornito come argomento.
- `getElementsByTagName()`: restituisce gli elementi che sono del tag fornito come parametro.

Esempio:

```
<p id="par123">Ciao</p>
<p name="abstract">Mondo!</p>
<button class="classeBella btn">Invia</button>
<a name="abstract" href="http://...">Link</a>
```



Attraversamento del DOM

Una volta ottenuto un elemento si può attraversare il DOM come albero, selezionando:

- Il nodo genitore:

```
let parent = element.parentNode;  
let parent = element.parentElement;
```

- I nodi figli:

```
let parent = element.childNodes;  
let parent = element.children;  
let first = element.firstChild; // element.firstChild;  
let last = element.lastChild; // element.lastElementChild;
```

- I nodi vicini:

```
let prev = element.previousSibling; //  
↪ element.previousElementSibling;  
let next = element.nextSibling; // element.nextElementSibling;
```



Selettori CSS (pt. 1)

Le interfacce `Document` e `Element` espongono due metodi che accettano un selettore CSS come argomento:

- `querySelector()`: restituisce il primo elemento che corrisponde al selettore.
- `querySelectorAll()`: restituisce una `NodeList`, cioè una lista, di elementi che corrispondono al selettore.

Differenza dei metodi tra `Document` e `Element`?

I metodi `Document` cercano gli elementi a partire dal nodo root del DOM, i metodi `Element` solo gli elementi che discendenti nel DOM a partire dall'elemento.

Selettori CSS (pt. 2)

I selettori CSS sono un **mini-linguaggio per definire pattern** che identificano uno o più elementi all'interno del DOM. Usati in origine solo in CSS, funzionano anche in JavaScript per selezionare elementi su cui poi poter operare.

- **Vantaggi:** rappresentazione compatta e rapida dei pattern. Si possono descrivere pattern molto complessi in modo facile rispetto ai metodi classici in JavaScript.
- **Svantaggi:** linguaggio esteso e complesso da imparare.

Selettori CSS (pt. 3)

Selettori base:

- **Per tipo:** seleziona gli elementi di una specifica tipologia.
"p" seleziona tutti gli elementi `<p>`.
- **Per classe:** seleziona gli elementi che hanno l'attributo `class` con il valore specificato.
".MyClass" seleziona tutti gli elementi con l'attributo `class="MyClass"`.
- **Per ID:** seleziona l'elemento che ha l'attributo `id` con il valore specificato.
"#MyID" seleziona l'elemento che ha l'attributo `id="MyID"`
- **Per attributo:** seleziona gli elementi con l'attributo specificato.
"[hidden]" seleziona tutti gli elementi che hanno l'attributo `hidden` con qualsiasi valore.



Selettori CSS (pt. 4)

- I selettori CSS possono essere combinati!

```
element.querySelector("a");  
document.querySelectorAll("p.MyApp");  
document.querySelectorAll(".MyApp");  
element.querySelectorAll("#myid");  
document.querySelectorAll("p");  
element.querySelector("button.classeBella#idUniVoCo")
```

- `querySelector()` restituisce `null` se non trova l'elemento,
`querySelectorAll()` restituisce una lista vuota.



2. Selezione degli elementi

Esercizio B

Consegna (pt. 1)

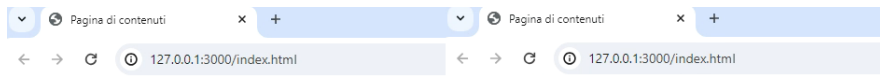
Partendo dallo scheletro su e-Learning, aggiungere codice JavaScript che modifichi la pagina nel seguente modo:

- 1 Correggere la prima intestazione `<h2>` in modo da visualizzare *Questa è una lista*.
- 2 Rimuovere l'elemento `Chrome` dalla lista.
- 3 Aggiungere una didascalia sotto la tabella.
- 4 Correggere l'intestazione della tabella utilizzando i tag HTML appropriati (`<thead>` e `<th>` per l'intestazione, `<tbody>` per il corpo della tabella).
- 5 Aggiungere un bordo alla tabella.
- 6 Rimuovere l'attributo `hidden` all'ultimo paragrafo.
- 7 Aggiungere una colonna alla tabella che indichi l'età, 22 per Mario e 20 per Alberto.
- 8 Aggiungere l'elemento `Haiku` alla lista.
- 9 Correggere il refuso *Weebbe* nel penultimo paragrafo.



Consegna (pt. 2)

Il risultato deve essere uguale al seguente (prima e dopo):



Pagina di contenuti Web

In questa pagina ci sono due elementi principali: una lista e una tabella.

Questa è una tabella

- Linux
- Windows
- MacOS
- Chrome

Questa è una tabella

Nome	Cognome	Numero
Mario	Rossi	+3395423853
Alberto	Gilardi	+332424532

Fine dei contenuti Weebble!

PRIMA

Pagina di contenuti Web

In questa pagina ci sono due elementi principali: una lista e una tabella.

Questa è una lista

- Linux
- Windows
- MacOS
- Haiku

Questa è una tabella

Nome	Cognome	Numero	Età
Mario	Rossi	+3395423853	22
Alberto	Gilardi	+332424532	20

Utenti del sistema.

DOPO

Fine dei contenuti Web!

Questo testo si dovrebbe vedere!



Suggerimenti

- Per sostituire dei nodi si possono usare i metodi:
 - `replaceChildren()`: sostituisce tutti i figli di un nodo,
 - `replaceWith()`: sostituisce il nodo con un altro.
- Per rimuovere:
 - `remove()`: rimuove il nodo dal DOM,
 - `removeAttribute()`: rimuove un attributo da un nodo.
- Osservare bene le classi, i nomi e gli ID impostati agli elementi HTML della pagina! Facilitano la selezione!
- Provare sia i selettori CSS sia i metodi classici di selezione.



3. Gestione degli eventi

Pagine statiche vs dinamiche

- Una pagina Web con solo HTML e CSS è statica.
- JavaScript rende dinamiche le pagine Web: può modificare il DOM in modo programmatico.

Esempio: al caricamento della pagine JavaScript modifica il DOM.

- È possibile far eseguire in automatico dal browser del codice JavaScript in risposta a eventi (es. click su un elemento, mouse che si sposta, pressione di un tasto della tastiera...).

Terminologia

- A un qualsiasi elemento nel DOM può essere associato uno o più **event handler**: sono funzioni JavaScript che vengono eseguite quando avviene l'evento.
- L'associazione dell'event handler all'elemento per un certo tipo di evento si chiama **registrazione**.



Il metodo `addEventListener()`

Tutti gli elementi del DOM espongono il metodo `addEventListener()` che permette di registrare una funzione a un evento per un elemento.

Parametri più importanti:

- ❶ `type`: una stringa che rappresenta l'evento da ascoltare.
- ❷ `listener`: funzione da invocare allo scatenarsi dell'evento. La funzione può avere un argomento opzionale di tipo `Event` che fornisce dettagli sull'evento avvenuto.

Il metodo `removeEventListener` rimuove la registrazione.

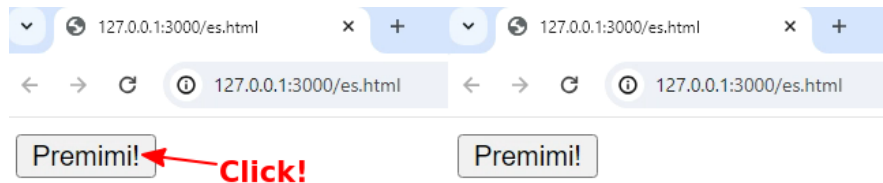


Esempio (pt. 1)

```
1  <html>
2    <body>
3      <button>Premimi!</button>
4      <script>
5        function buttonPress() {
6          const p = document.createElement("p");
7          p.appendChild(document.createTextNode("Hai premuto il
8            ↪ pulsante!"));
9          document.body.appendChild(p);
10        }
11
12        window.onload = init;
13
14        function init() {
15          const button = document.querySelector("button");
16          button.addEventListener("click", buttonPress);
17        };
18      </script>
19    </body>
20  </html>
```



Esempio (pt. 2)



Hai premuto il pulsante!

Osservazioni

- Si può aggiungere più di un *event handler* per uno stesso evento:

```
1 element.addEventListener("click", function1);  
2 element.addEventListener("click", function2);
```

Le funzioni verranno eseguite in ordine.

- Non mischiare JavaScript e HTML: per registrare un evento è meglio usare `querySelector()` seguito da `addEventListener()` che usare gli attributi come `onclick` per un pulsante.
- Come trovare gli eventi? Ogni elemento ha il suo elenco. Per esempio `HTMLFormElement` (`<form>`) ha gli eventi `submit` e `reset`.



Prevenire il comportamento predefinito

Alcuni eventi associati ai tag HTML sono predefiniti. Per esempio l'azione di invio di un form HTML (`<input type="submit" value="Submit" />`).

Per poter intercettare tali eventi, o addirittura annullarli, è necessario invocare il metodo `preventDefault()` dell'interfaccia `Event`.

Per esempio, ipotizzando che ci sia un form HTML con un pulsante di invio, il seguente codice JavaScript impedisce l'invio dei dati:

```
1  // "form" e "p" sono restituiti da querySelector().
2  form.addEventListener("submit", myFunction);
3
4  function myFunction(event) {
5      event.preventDefault();
6      p.textContent = "Non puoi inviare il form!";
7  }
```



3. Gestione degli eventi

Esercizio C

Consegna (pt. 1)

Partendo dallo scheletro su e-Learning, aggiungere codice JavaScript che altera il comportamento della pagina:

- Al clic sul pulsante di invio del form viene aggiunta la tastiera nella tabella e pulisce i campi del form.
- Per ogni riga della tabella ci deve essere un pulsante che permetta di rimuovere l'intera riga.

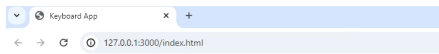
Attenzione

- ❶ Non si può modificare il codice HTML, solo aggiungere codice JavaScript.
- ❷ Il form **NON** deve inviare la richiesta POST al server.



Consegna (pt. 2)

Il risultato deve essere uguale al seguente:



Keyboard App

List of saved keyboards

Name	Manufacturer	Year	Ergonomic	Backlight	Layout	
ThinkKeyboard	Lenovo	2012	false	true	QWERTY	Remove

Add a new keyboard

Name:

Manufacturer:

Year:

Ergonomic:

☐

Backlight:

☐

Layout:



Suggerimenti

- Quando si seleziona un tag HTML `input` viene restituito un oggetto di tipo `HTMLInputElement`.
Occhio alle proprietà! In particolare `value` e `checked`, tornano utili per ottenere i dati di un form in JavaScript.
- Quando si genera la riga e in particolare il pulsante per rimuoverla, si possono aggiungere eventi al pulsante appena creato.
Nella funzione che gestirà l'eliminazione della riga, accettare l'argomento di tipo `Event` ed esplorare le proprietà che contiene, tra cui `currentTarget` che torna utile nell'handler per sapere su quale elemento HTML è avvenuto l'evento.