





FORM



Form

Ricevere dati dall'utente

1 Input Text

2 Input Email

3 Textarea

4 Checkbox

5 Button

Nome*

Inserisci il tuo nome 1

Email

Inserisci la tua email 2

Messaggio*

Messaggio 3

4 ☐ Accetto la Privacy policy

INVIA 5



Form

Ricevere dati dall'utente

Nome*

Inserisci il tuo nome

Email

Inserisci la tua email

Messaggio*

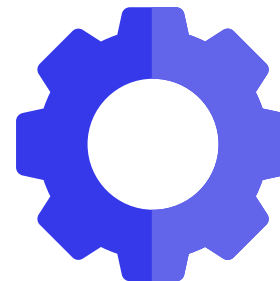
Messaggio

☐ Accetto la Privacy policy

INVIA

Al click del tasto INVIA, tutto il contenuto del form viene inviato ad una parte del backend che svolgerà le opportune azioni.

In questo caso, salvare le informazioni sul database e inviare una email.





Form

Input tag

```
1 <!-- <input type="tipo"> -->
2
3 <input type="text">
4 <input type="number">
5 <input type="color">
6 <input type="checkbox">
7 <input type="radio">
8 <input type="date">
9 <input type="file">
10 <input type="month">
11 <input type="password">
12 <input type="range">
13 <input type="time">
14 <input type="submit">
```

A visual representation of an HTML form with various input types. The form is displayed in a window with a white background and a light gray border. It includes the following elements:

- A single-line text input field at the top.
- A row of controls: a spinner (up/down arrows), a color picker (black square), two radio buttons (one selected, one unselected), and a date input field showing "mm/dd/yyyy".
- A file upload section: a button labeled "Choose File", the text "No file chosen", and a text input field with dashes.
- A range input: a text input field followed by a horizontal slider.
- A time input: a text input field showing "--:--" and a button labeled "Submit".



Form

Input tag

```
1 <!-- <input type="tipo"> -->
2
3 <input type="text">
4 <input type="number">
5 <input type="color">
6 <input type="checkbox">
7 <input type="radio">
8 <input type="date">
9 <input type="file">
10 <input type="month">
11 <input type="password">
12 <input type="range">
13 <input type="time">
14 <input type="submit">
```

A visual representation of an HTML form with various input types. The form is displayed in a window with a title bar containing three colored buttons (red, yellow, green). The form elements include:

- A single-line text input field.
- A dropdown menu (select) showing a downward arrow.
- A color input field showing a black color swatch.
- A checkbox input field.
- A radio button input field.
- A date input field showing the format "mm/dd/yyyy".
- A file input field showing a "Choose File" button and the text "No file chosen".
- A month input field showing a calendar icon.
- A range input field showing a slider.
- A time input field showing a clock icon.
- A submit button labeled "Submit".



Tipi di Input

Select

```
1 <select name="scelta">
2   <option value="opt1">Opzione1</option>
3   <option value="opt2">Opzione2</option>
4 </select>
5
```



A browser window mockup with a white background and three colored window control buttons (red, yellow, green) in the top-left corner. Inside the window, there is a single HTML select element. The select box is a rounded rectangle with a thin black border. It contains the text 'Opzione1' in a black sans-serif font, followed by a black downward-pointing triangle icon on the right side, indicating a dropdown menu.



Tipi di Input

Textarea

```
1 <textarea name="name" rows="8" cols="80">  
2 </textarea>  
3
```



Possiamo inserire un testo più lungo disposto su più righe



Form Validation

HTML5 identifica alcune parole chiave per validare il contenuto dei form

```
1 <input type="tipo" required>
```

Di default, i browser controllano che ci sia corrispondenza tra il "type" dichiarato e ciò che viene inserito, `required` obbliga la presenza di un valore, `minlength`, `maxlength` forzano un numero minimo e un numero massimo di caratteri (o ancora meglio l'attributo "pattern").

In caso il contenuto non sia valido, il form non viene inviato.

Queste regole sono specifiche di HTML5. Non funzionano su browser più datati e sono più facilmente aggirabili, quindi alla client-side validation va sempre affiancata la server-side validation (es. live)



LIVE CODING



Form e JavaScript

Aka: Come lavorare coi campi dei form



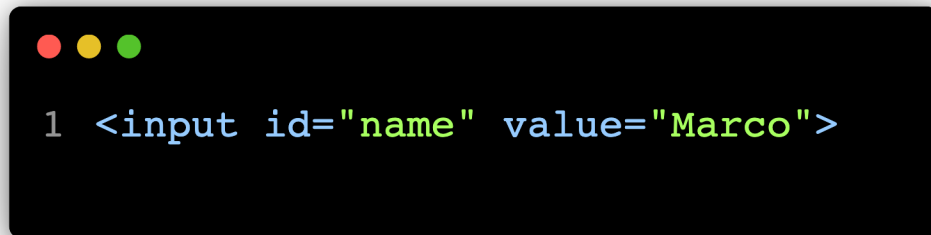
Form e JS

Leggere il contenuto dei campi di un form

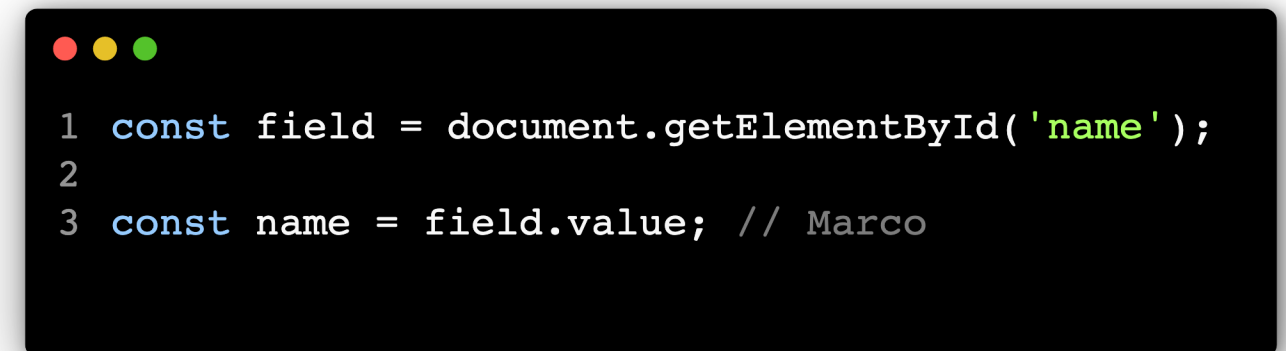
Tutti i tag tipici dei form (come `<input>`, `<select>`, `<textarea>`, ecc.) ci danno la possibilità di leggere il valore al loro interno tramite l'attributo **value**.



A screenshot of a web browser window showing a simple form. The form has a label "Nome:" followed by a text input field containing the text "Marco". Below the input field is a button labeled "Invia".



```
1 <input id="name" value="Marco">
```



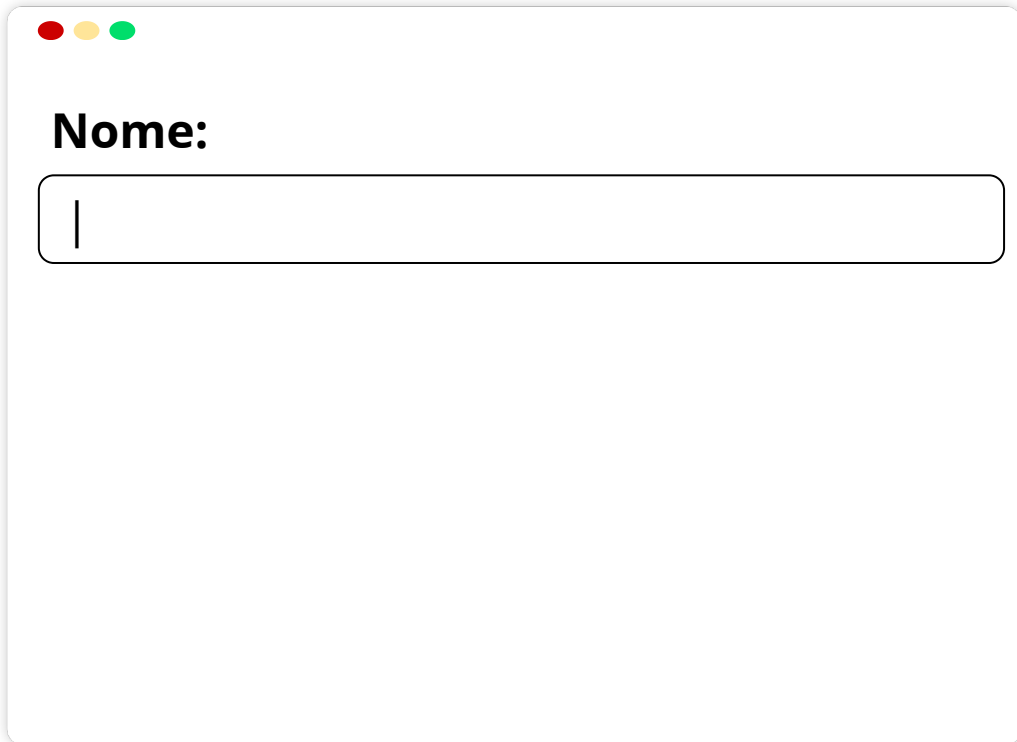
```
1 const field = document.getElementById('name');  
2  
3 const name = field.value; // Marco
```



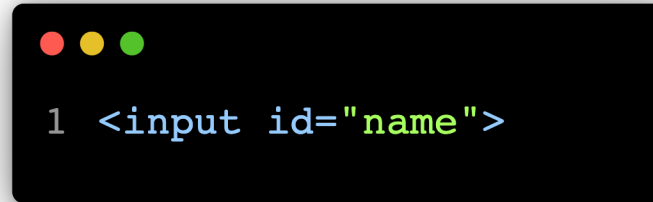
Form e JS

Leggere il contenuto dei campi di un form

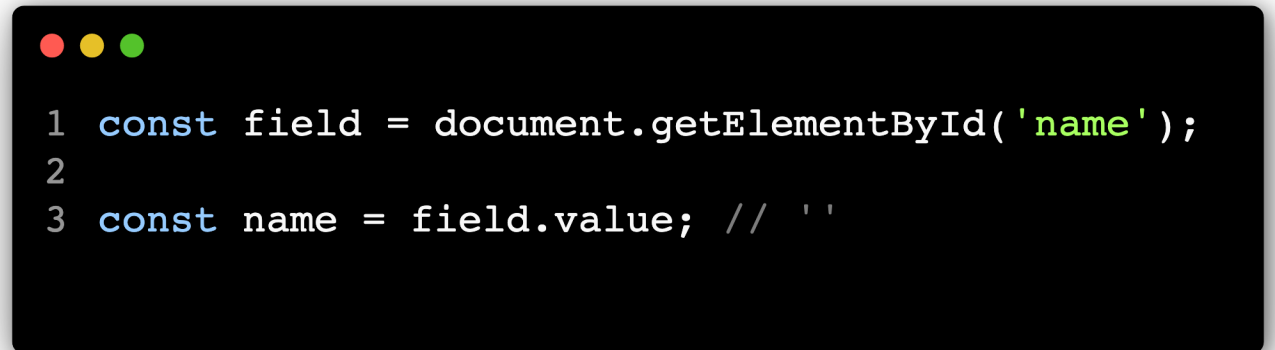
Il problema è che all'avvio della pagina, i nostri campi sono quasi sempre vuoti, perchè l'utente deve ancora scriverci dentro! Di conseguenza, **eseguire questo codice darebbe sempre stringa vuota!**



Nome:



```
1 <input id="name">
```



```
1 const field = document.getElementById('name');  
2  
3 const name = field.value; // ''
```



Form e JS

Leggere il contenuto dei campi di un form

Fortunatamente, a noi interessa leggere il contenuto solo dopo che l'utente invia il form, pertanto possiamo sfruttare un button e un **eventListener**!



```
1 <input id="name" value="Marco">
2 <button>Invia</button>
```



Nome:



```
1 const field = document.getElementById('name');
2 const button = document.querySelector('button');
3
4 // se proviamo a leggere qui, sarà sempre vuoto
5 const name = field.value; //
6
7 button.addEventListener('click', () => {
8   const name = field.value; // Marco
9 });
```



LIVE CODING



Form

E il <form>?

Tutti gli input che vogliono essere inviati devono essere incapsulati in un tag **<form>**

Solo gli input che sono all'interno vengono inviati all'indirizzo definito in **action**.

Se il valore di **action** è vuoto, la pagina invia i dati a se stessa e si ricarica.

Anche il button deve essere all'interno del form affinché ne invii i campi quando lo clicchiamo.

```
1 <form action="www.miosito.com/boh...">
2
3   <input type="text">
4   <button>Send Me</button>
5
6 </form>
7
```




Form

Il Tag `<form>` e il suo comportamento naturale

Quando il browser capisce che stiamo inviando un form, a prescindere dal valore di action, effettua una nuova richiesta e quindi **perdiamo tutti i dati JavaScript**, come se avessimo aggiornato la pagina.

Questo succede perchè normalmente **i form sono nati per inviare dati da una pagina a un'altra**. E spesso l'altra pagina è una pagina di backend.

Tuttavia, **noi vogliamo gestire i dati con JavaScript**, restando nella nostra pagina.. come possiamo fare?



Form

Giù le mani dal mio form!

La soluzione consiste nell'intercettare l'evento di invio del button (evento **submit**) e impedire che effettui il suo comportamento di default!

Sfruttiamo un event listener all'evento **submit** del form.

Sfruttiamo il parametro **event**, che ci viene fornito "magicamente" da JS.

Blocciamo il comportamento naturale con **event.preventDefault()**



```
1 // Recupero il form
2 const form = document.querySelector('form');
3
4 // Intercetto l'evento di invio
5 form.addEventListener('submit', (event) => {
6   // Blocco l'invio del form..
7   event.preventDefault();
8
9   // Tutto il resto del codice...
10 });
```



LIVE CODING



ESERCITAZIONE