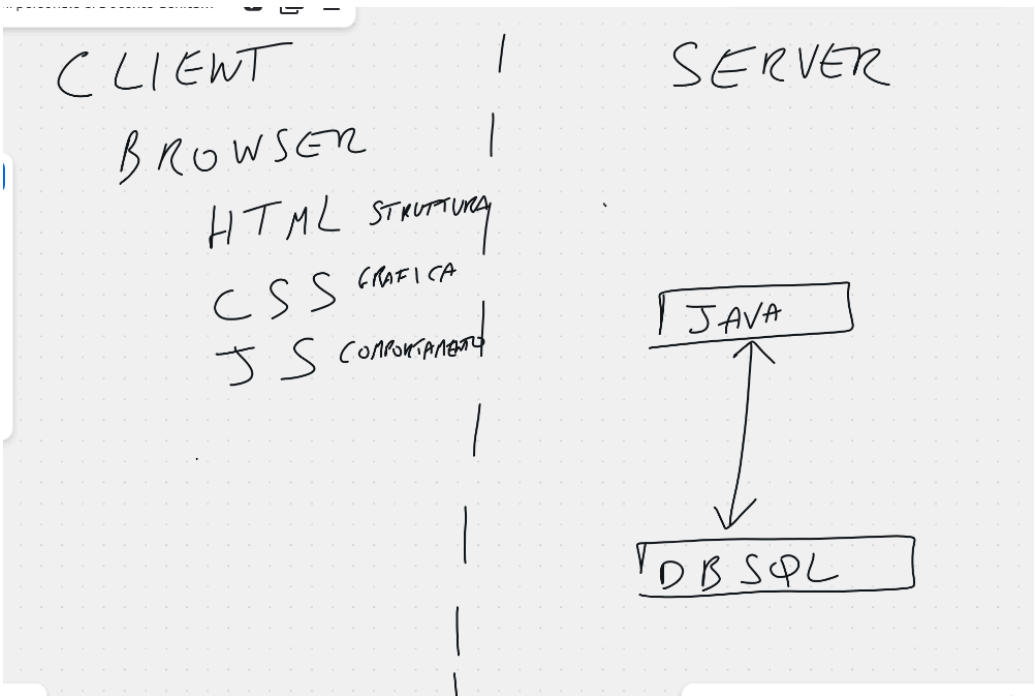
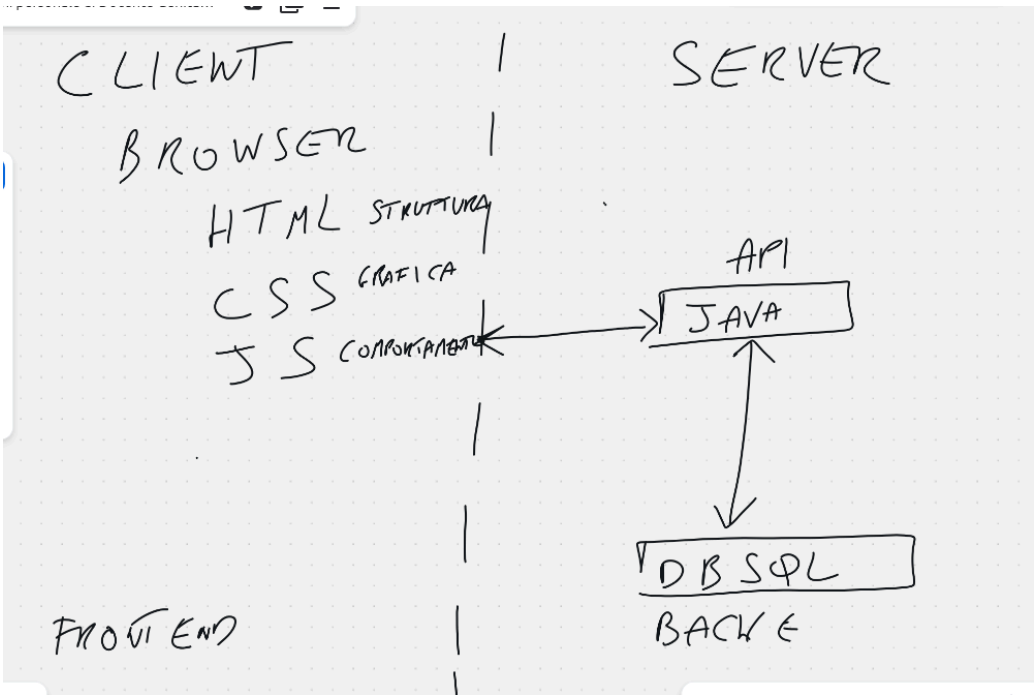
Appunti 03-02

L'HTML è nato originariamente con lo scopo di mostrare dati scientifici, ma nel corso del tempo ha assunto un ruolo molto più ampio. HTML non è un linguaggio di programmazione, bensì un linguaggio per scrivere ipertesti, ovvero testi che possono contenere non solo testo, ma anche immagini, animazioni e musica. La struttura di una pagina HTML è arricchita da altri due linguaggi che lavorano al suo interno. Il CSS rappresenta un secondo linguaggio incorporato nell'HTML, paragonabile a un mitocondrio, che non fornisce la struttura della pagina ma si occupa della sua grafica e del suo aspetto visivo. Il JavaScript è invece un vero linguaggio di programmazione che viene eseguito direttamente nel browser e conferisce il comportamento alla pagina. Grazie a questa integrazione, una pagina web non è più semplicemente un documento statico, ma può trasformarsi in un vero e proprio programma. L'HTML nasce dunque come documento, ma attraverso l'integrazione con CSS e JavaScript può evolvere e diventare un programma interattivo



Sul server sono presenti le API, che consistono in un programma Java capace di comunicare con un database SQL. Questa separazione tra tecnologie diverse

risponde a vincoli tecnici ben precisi, poiché le competenze e gli strumenti necessari per gestire un database non coincidono con quelli richiesti per scrivere un programma applicativo. Scrivere un programma completo direttamente in SQL risulterebbe infatti impossibile o estremamente limitante. Dal lato client non troviamo Java per ragioni storiche legate all'evoluzione del web. Ciò che stiamo osservando rappresenta l'evoluzione delle tecnologie nel mondo del lavoro. Java è stato adottato perché serviva fornire un comportamento dinamico alle pagine web e, essendo già disponibile e maturo, è stato integrato in questo contesto. Tuttavia non esiste ancora uno standard definitivo e consolidato, e nel corso degli anni le tecnologie continueranno inevitabilmente a cambiare. Per questo motivo è fondamentale imparare a padroneggiare la logica di programmazione piuttosto che concentrarsi esclusivamente su uno strumento specifico, dato che lo strumento potrebbe essere sostituito o evolversi nel tempo.



Sul server sono presenti le API, che consistono in un programma Java capace di comunicare con un database SQL.

UN PO DI FRONTEND

Per quanto riguarda il frontend, HTML è un linguaggio di markup la cui struttura si basa su un principio fondamentale. Un file HTML è essenzialmente composto da scatole dentro scatole dentro scatole, dove ogni elemento può contenere altri elementi in una gerarchia annidata che definisce l'intera struttura del documento.

I tag in HTML sono i mattoni fondamentali con cui si costruisce la struttura di una pagina web.

Ogni tag è racchiuso tra parentesi angolari, per esempio `<p>` per un paragrafo o `<h1>` per un titolo principale.

Quasi tutti i tag hanno una forma di apertura e una di chiusura, come `<div>` e `</div>`, che racchiudono il contenuto a cui si applicano.

Esistono tag che descrivono la struttura della pagina, come `<header>`, `<main>` e `<footer>`.

Altri tag servono per il contenuto testuale, come `<p>`, `<span>`, `<strong>` e `<em>`.

Ci sono tag specifici per le immagini e i media, ad esempio `<img>` per le immagini e `<video>` per i video.

Alcuni tag, detti “vuoti” o “self-closing”, non hanno contenuto interno, come `<br>` o `<hr>`.

I tag possono avere attributi, cioè coppie nome-valore che aggiungono informazioni o comportamento, come `class`, `id` o `href`.

L’insieme dei tag e dei loro attributi definisce la semantica del documento, aiutando sia il browser sia i motori di ricerca a capire il contenuto.

Scrivere correttamente i tag, rispettando la gerarchia e la chiusura, è essenziale per ottenere pagine web accessibili, mantenibili e visualizzate in modo coerente dai browser.

Il carattere

```
/
```

in HTML si usa principalmente per indicare la chiusura di un tag, per esempio

```
</p>
```

chiude il paragrafo aperto da

```
<p>
```

Nei tag di chiusura il **slash** dice al browser dove termina l'elemento, così può distinguere correttamente le varie parti della pagina

Per andare a capo in HTML si utilizza il tag `<br>`, un elemento vuoto che forza una interruzione di linea senza creare un nuovo paragrafo.geeksforgeeks+1

A differenza di altri tag, `<br>` non richiede una chiusura con `</br>` poiché è autochiudente, e può essere usato più volte consecutive per spaziatura verticale.developer.mozilla+1

Il tag `<i>` in HTML serve a inserire testo in corsivo, indicando enfasi o stile italico per una porzione di contenuto

Viene utilizzato racchiudendo il testo desiderato tra `<i>` e `</i>.`

HTML permette di raccogliere informazioni direttamente dall'utente, fungendo da equivalente della console in un ambiente interattivo.

In questo modo si crea un'interazione diretta con l'utente attraverso elementi specifici della pagina.

Il tag `<form>` rappresenta un elemento particolare che consente di ricevere dati inseriti dall'utente in modo strutturato.

Una pagina HTML diventa così uno strumento bidirezionale di input e output: dalla pagina si possono leggere i risultati visualizzati, mentre alla pagina si possono inviare valori che l'utente ha compilato.

Il tag `<input>` è l'elemento fondamentale per creare controlli interattivi in un form HTML e raccogliere dati dall'utente.

L'attributo `type` definisce la natura specifica dell'input, trasformando radicalmente il modo in cui il browser visualizza e gestisce il controllo. Ad esempio, `type="text"` crea una casella di testo semplice, `type="password"` nasconde i caratteri digitati, mentre `type="radio"` o `type="checkbox"` creano opzioni selezionabili

L'attributo `name` è invece cruciale per l'invio dei dati: assegna un identificativo al campo che il server utilizzerà per recuperare il valore inserito. Senza l'attributo `name`, il dato inserito nell'input non verrebbe trasmesso al momento del submit del form.

L'attributo

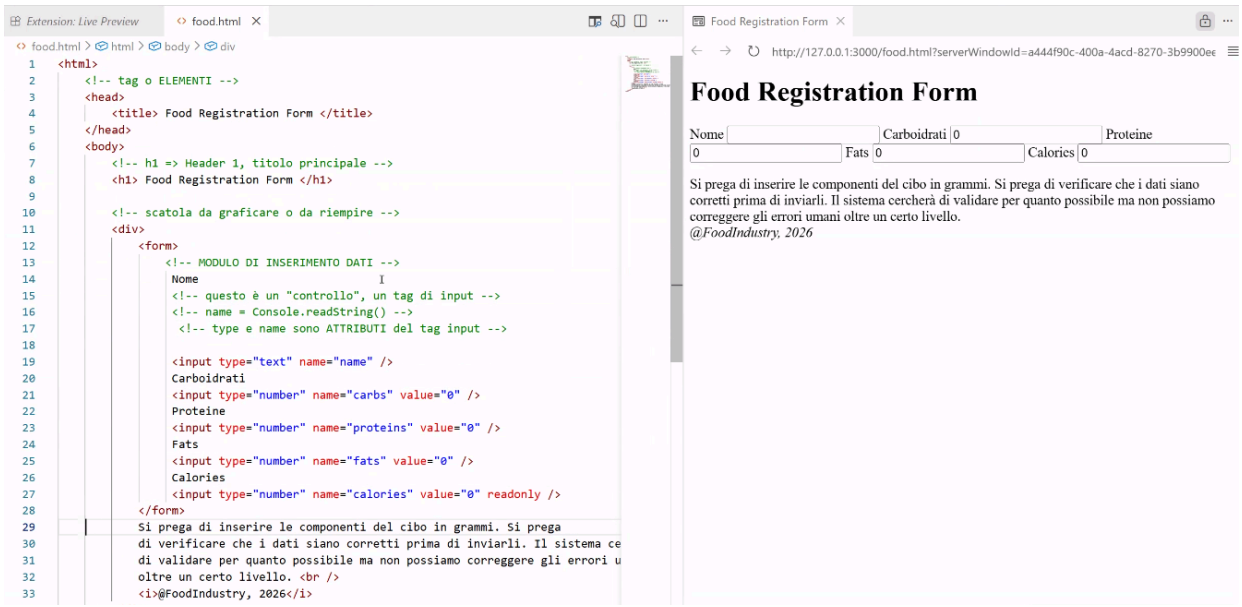
```
value="0"
```

imposta un valore iniziale predefinito per i campi numerici, rendendoli pronti all'uso senza che l'utente debba azzerarli manualmente.

Infine

```
readonly
```

sul campo "Calories" impedisce la modifica diretta da parte dell'utente, suggerendo che il valore sarà calcolato automaticamente dal sistema in base agli altri input.



All'interno del tag `<head>` si apre un tag chiamato `<style>`, che introduce il mondo di CSS.

Si entra così in un linguaggio completamente diverso da HTML, con regole proprie e sintassi distinta, dedicato esclusivamente alla grafica degli elementi della pagina.

CSS è composto da regole di grafica che definiscono aspetto, colori, dimensioni, posizioni e layout di tutti gli elementi HTML.

Queste regole di CSS potranno essere scritte in vari contesti: direttamente negli elementi HTML, in file esterni, nel `<head>`, o addirittura integrate nel JavaScript.

Il CSS si applica tramite selettori che puntano agli elementi HTML, come nell'esempio che mostra `body, h1, div` per applicare lo stile a più tag contemporaneamente.

Ogni regola CSS inizia con un selettore seguito dalle graffe `{}` che contengono le proprietà da applicare.

Le proprietà sono scritte come coppie nome-valore, separate da punto e virgola, come `font-family: Arial;`.

Il selettore `body, h1, div` indica che lo stile verrà applicato a tutti e tre gli elementi elencati.

`font-family` è una proprietà che definisce il tipo di carattere da utilizzare per il testo.

Il valore `'Arial'` specifica il font desiderato, racchiuso tra virgolette per garantire la compatibilità.

Le graffe `{}` racchiudono tutte le proprietà di una regola CSS, terminando con `}` per chiuderla.

Questa sintassi di base permette di controllare aspetto, layout e comportamento visivo di qualsiasi elemento HTML.

I selettori possono essere multipli, separati da virgole, per applicare lo stesso stile a gruppi di elementi.

border

è una proprietà CSS fondamentale che definisce il contorno di un elemento su tutti i lati. Può essere specificata con tre valori: border: 2px solid black; dove 2px è lo spessore, solid lo stile e black il colore.

Esistono proprietà singole come

- border-top
- border-right
- border-left
- border-bottom

per controllare lati specifici.

Gli stili disponibili includono

- solid (linea continua),
- dashed (tratteggiato),
- dotted (punteggiato) e
- double (doppia linea).

border aiuta a creare separazioni visive, decorazioni e struttura per bottoni, tabelle e contenitori nella pagina.

Proprietà	Descrizione	Esempio
`font-family`	Definisce il tipo di carattere	`font-family: 'Arial';`

Proprietà	Descrizione	Esempio
`font-size`	Imposta la dimensione del testo	`font-size: 16px;`
`color`	Colore del testo	`color: black;`
`margin`	Spazio esterno su tutti i lati	`margin: 10px;`
`margin-top`	Margine superiore	`margin-top: 20px;`
`padding`	Spazio interno su tutti i lati	`padding: 15px;`
`border`	Contorno completo	`border: 2px solid black;`
`border-left`	Bordo sinistro	`border-left: 1px dashed gray;`
`background-color`	Colore di sfondo	`background-color: white;`

È possibile assegnare agli elementi HTML degli identificativi univoci tramite l'attributo `id`. In CSS si possono quindi creare regole specifiche per questi identificativi precedendo il loro nome con il simbolo `#`.

L'attributo `id` è un identificativo univoco assegnato a un singolo elemento HTML per selezionarlo in modo specifico.

Viene utilizzato sia in CSS, preceduto dal simbolo `#` (es. `#menu`), sia in JavaScript per manipolare quell'elemento preciso.

Ogni pagina deve avere ID univoci, senza duplicati, per garantire che i selettori funzionino correttamente e senza conflitti.

JavaScript e Interazione Dinamica con le Pagine Web

A questo punto si introduce l'altro componente fondamentale dell'ecosistema web: JavaScript. JavaScript opera basandosi sugli eventi, dove un evento rappresenta qualunque cosa che accade alla pagina. Può trattarsi di un click del mouse, della pressione di un tasto sulla tastiera, del caricamento completo della pagina, della modifica del valore di un campo input, o di qualsiasi altra interazione dell'utente con gli elementi della pagina.

Per comodità, il primo passo consiste nell'assegnare un nome alla form, ad esempio "food", che risulterà utile successivamente per accedere ai dati contenuti

al suo interno. Questo nome funziona come un identificatore che JavaScript può utilizzare per riferirsi all'intera form e a tutti i suoi elementi figlio.

CSS non possiede capacità di calcolo. CSS è un linguaggio puramente dichiarativo dedicato alla presentazione visuale, mentre è necessario eseguire espressioni matematiche per implementare logica di business. Ogni volta che cambiano i valori di carboidrati, proteine o grassi inseriti dall'utente, devono essere ricalcolate le calorie attraverso un'espressione specifica

In JavaScript ogni elemento della pagina diventa un oggetto. Questa è la differenza fondamentale di paradigma tra HTML e JavaScript: quello che in HTML è rappresentato da tag markup, in JavaScript diventa un oggetto con proprietà e metodi. JavaScript ha accesso diretto agli elementi della pagina attraverso il Document Object Model, il DOM, che per l'HTML sono tag mentre per JavaScript sono oggetti manipolabili programmaticamente.

JavaScript presenta tipizzazione debole e non è tipizzato staticamente, poiché il tipo di una variabile è determinato dal suo contenuto senza possibilità di dichiarare tipi espliciti al momento della definizione. Una variabile può contenere un numero in un momento e successivamente una stringa senza generare errori di compilazione. Questo contrasta radicalmente con linguaggi come Java dove il tipo viene dichiarato esplicitamente e non può cambiare durante l'esecuzione.

È possibile accedere agli attributi dei tag in JavaScript tramite il punto, trasformandoli così in proprietà dell'oggetto corrispondente. Se un elemento HTML ha un attributo value, in JavaScript questo diventa accessibile come oggetto.value. Gli attributi HTML class, id, name e tutti gli altri diventano proprietà JavaScript che possono essere lette e modificate dinamicamente, permettendo di alterare sia il contenuto che l'aspetto della pagina in risposta alle azioni dell'utente.

Questo meccanismo di trasformazione degli elementi HTML in oggetti JavaScript manipolabili costituisce il fondamento dell'interattività delle applicazioni web moderne. JavaScript può leggere i valori inseriti dall'utente nei campi input, eseguire calcoli su questi valori, e aggiornare immediatamente altri elementi della pagina con i risultati, tutto senza necessità di ricaricare la pagina o comunicare con il server.

Event Handling e Funzioni JavaScript

Per associare un comportamento di una funzione JavaScript a un evento che si verifica sulla pagina, si utilizza l'attributo HTML onChange con il valore uguale al nome della funzione da eseguire, ad esempio onChange="updateCalories()". Questo gestore di eventi viene inserito direttamente nel tag HTML corrispondente, tipicamente un elemento input, select o textarea che può essere modificato dall'utente.

updateCalories è la funzione che gestisce specificamente l'evento change, ovvero il momento in cui l'utente modifica i valori di carboidrati, proteine o grassi nei rispettivi campi di input. L'evento change si attiva quando l'utente cambia il contenuto di un campo e poi sposta il focus altrove, segnalando che la modifica è completa.

Quando si verifica questo evento, il browser esegue automaticamente il metodo updateCalories per ricalcolare le calorie totali. Questo meccanismo crea un collegamento diretto tra l'azione dell'utente sulla pagina e l'esecuzione di codice JavaScript, permettendo alla pagina di rispondere immediatamente alle interazioni senza necessità di ricaricare o inviare dati al server.

Il pattern è semplice ma potente: l'attributo HTML onChange funge da ponte tra il DOM e il codice JavaScript, specificando quale funzione deve essere invocata quando l'evento specifico si verifica. All'interno della funzione updateCalories, il codice JavaScript accede ai valori correnti dei campi carboidrati, proteine e grassi, applica le formule di calcolo appropriate moltiplicando ciascun macronutriente per il suo coefficiente calorico, somma i risultati e aggiorna dinamicamente il valore visualizzato nel campo delle calorie totali.

Questo approccio di gestione degli eventi inline, dove il gestore è specificato direttamente nell'attributo HTML, rappresenta uno dei metodi più semplici e diretti per collegare eventi a funzioni JavaScript, particolarmente adatto per applicazioni di complessità contenuta o per scopi didattici dove la chiarezza della relazione tra elemento e comportamento è prioritaria.

Event Handling, Target e Change Detection

I dettagli dell'evento, come il target, sono fondamentali per capire dove si verifica l'azione e quale elemento specifico ha generato l'evento. L'evento in questione è change, mentre i target sono i campi input per carboidrati, proteine e grassi. Ogni campo input che ha l'attributo onChange="updateCalories()" diventa un potenziale target per questo evento.

La risposta a questo evento è l'esecuzione della funzione updateCalories, che viene chiamata ogni singola volta che uno di questi elementi subisce una modifica. Non importa quale dei tre campi venga modificato, la stessa funzione viene invocata per ricalcolare il totale delle calorie basandosi sui valori correnti di tutti e tre i macronutrienti.

JavaScript mantiene una sorveglianza costante su tutti gli oggetti della pagina, attivando il metodo ogni qualvolta si verifica l'evento specifico per cui è stato registrato un gestore. Questo monitoraggio continuo avviene attraverso il sistema di gestione degli eventi del browser, che tiene traccia di tutti i listener registrati e dei loro elementi associati.

Esiste infatti un sistema di change detection che si basa su un meccanismo chiamato observer, il quale monitora continuamente le modifiche allo stato degli elementi. Questo pattern observer è fondamentale nell'architettura delle applicazioni web moderne: quando lo stato di un elemento cambia, tutti i listener registrati per quell'evento specifico vengono notificati e i relativi gestori vengono eseguiti automaticamente.

Il browser mantiene internamente una mappatura tra elementi del DOM, tipi di eventi e funzioni handler da eseguire. Quando l'utente interagisce con la pagina modificando il valore di un campo input e spostando poi il focus altrove, il browser rileva questo cambiamento di stato, identifica quale evento è stato generato, determina quale elemento è il target, e quindi esegue tutte le funzioni registrate per gestire quell'evento su quell'elemento specifico.

Questo meccanismo di change detection basato su observer permette alla pagina di rimanere reattiva e aggiornata in tempo reale senza necessità di polling continuo o ricaricamenti della pagina. JavaScript non deve verificare manualmente e ripetutamente se qualcosa è cambiato: il browser notifica automaticamente il codice JavaScript quando gli eventi rilevanti si verificano, garantendo efficienza ed efficacia nella gestione delle interazioni utente.

Event Handling, Target e Change Detection

I dettagli dell'evento, come il target, sono fondamentali per capire dove si verifica l'azione e quale elemento specifico ha generato l'evento. L'evento in questione è change, mentre i target sono i campi input per carboidrati, proteine e grassi. Ogni campo input che ha l'attributo onChange="updateCalories()" diventa un potenziale target per questo evento.

La risposta a questo evento è l'esecuzione della funzione updateCalories, che viene chiamata ogni singola volta che uno di questi elementi subisce una modifica. Non importa quale dei tre campi venga modificato, la stessa funzione viene invocata per ricalcolare il totale delle calorie basandosi sui valori correnti di tutti e tre i macronutrienti.

JavaScript mantiene una sorveglianza costante su tutti gli oggetti della pagina, attivando il metodo ogni qualvolta si verifica l'evento specifico per cui è stato registrato un gestore. Questo monitoraggio continuo avviene attraverso il sistema di gestione degli eventi del browser, che tiene traccia di tutti i listener registrati e dei loro elementi associati.

Esiste infatti un sistema di change detection che si basa su un meccanismo chiamato observer, il quale monitora continuamente le modifiche allo stato degli elementi. Questo pattern observer è fondamentale nell'architettura delle applicazioni web moderne: quando lo stato di un elemento cambia, tutti i listener registrati per quell'evento specifico vengono notificati e i relativi gestori vengono eseguiti automaticamente.

Il browser mantiene internamente una mappatura tra elementi del DOM, tipi di eventi e funzioni handler da eseguire. Quando l'utente interagisce con la pagina modificando il valore di un campo input e spostando poi il focus altrove, il browser rileva questo cambiamento di stato, identifica quale evento è stato generato, determina quale elemento è il target, e quindi esegue tutte le funzioni registrate per gestire quell'evento su quell'elemento specifico.

Questo meccanismo di change detection basato su observer permette alla pagina di rimanere reattiva e aggiornata in tempo reale senza necessità di polling continuo o ricaricamenti della pagina. JavaScript non deve verificare manualmente e ripetutamente se qualcosa è cambiato: il browser notifica automaticamente il codice JavaScript quando gli eventi rilevanti si verificano,

garantendo efficienza ed efficacia nella gestione delle interazioni utente.Gestori di Eventi in HTML e JavaScript

onChange è uno dei tanti gestori di eventi disponibili in HTML e JavaScript per collegare azioni dell'utente a comportamenti programmati. Tra i più comuni e fondamentali ci sono onclick per i click del mouse, onSubmit per l'invio dei form, onLoad per il caricamento della pagina e onKeyDown per la pressione dei tasti. Questi gestori si associano direttamente agli elementi HTML tramite attributi, permettendo di eseguire codice JavaScript in risposta ad azioni dell'utente.

onclick si utilizza tipicamente sui pulsanti, link o qualsiasi elemento cliccabile della pagina. Quando l'utente fa click sull'elemento, la funzione JavaScript specificata viene eseguita immediatamente. Ad esempio, un pulsante con onclick="calcolaRisultato()" eseguirà quella funzione ogni volta che viene premuto.

onSubmit viene applicato specificamente all'elemento form e si attiva quando l'utente tenta di inviare il form, tipicamente premendo un pulsante di tipo submit o premendo Invio in un campo di input. Questo gestore è particolarmente utile per validare i dati inseriti dall'utente prima che vengano effettivamente inviati al server, permettendo di intercettare l'invio e prevenirlo se i dati non sono corretti.

onLoad si associa solitamente all'elemento body o window e si attiva quando la pagina ha completato il caricamento di tutti i suoi elementi, incluse immagini, script e fogli di stile. Questo gestore è fondamentale quando si necessita di eseguire codice JavaScript che dipende dalla presenza completa del DOM, garantendo che tutti gli elementi siano disponibili prima di tentare di manipolarli.

onKeyDown rileva il momento esatto in cui l'utente preme un tasto sulla tastiera, prima ancora che il carattere venga inserito nel campo. Questo evento è utile per intercettare combinazioni di tasti, implementare scorciatoie da tastiera o validare l'input in tempo reale mentre l'utente digita. Esiste anche onKeyUp che si attiva quando il tasto viene rilasciato, e onKeyPress che si attiva durante la pressione.

Altri gestori di eventi comunemente utilizzati includono onMouseOver che si attiva quando il cursore passa sopra un elemento, onMouseOut quando il cursore esce dall'elemento, onFocus quando un campo riceve il focus, onBlur quando lo perde, onInput che si attiva ad ogni modifica in tempo reale di un campo di input anche prima che l'utente completi l'inserimento, e onChange che invece si attiva solo quando il valore cambia e l'elemento perde il focus.

Questi gestori di eventi inline rappresentano il modo più diretto e visualmente chiaro di collegare eventi a comportamenti, rendendo immediatamente evidente quale funzione viene eseguita in risposta a quale azione dell'utente. Costituiscono il fondamento dell'interattività nelle applicazioni web, trasformando pagine statiche in interfacce dinamiche e responsive che reagiscono in tempo reale alle azioni degli utenti.