Progetto DR - Appunti — 17/02/2026

Un form HTML rappresenta una delle strutture fondamentali del web moderno: è il meccanismo attraverso cui un utente può inserire dati e trasmetterli verso un server. L'intera struttura ruota attorno al tag `<form>`, che funge da contenitore logico per tutti gli elementi di input al suo interno. Questo tag richiede due attributi fondamentali per funzionare correttamente: `action`, che definisce l'indirizzo del server destinatario dei dati, e `method`, che stabilisce il metodo HTTP con cui effettuare la trasmissione. In particolare, si usa `post` quando i dati da inviare sono sensibili, come nel caso di credenziali di accesso, mentre `get` è appropriato quando i dati possono essere resi visibili direttamente nell'URL.

L'elemento centrale all'interno di un form è il tag `<input>`, la cui natura è intrinsecamente polivalente: il suo comportamento cambia radicalmente in funzione del valore assegnato all'attributo `type`. Con `type="text"` si ottiene un campo di inserimento testuale generico, mentre `type="email"` attiva una validazione automatica lato browser che verifica la correttezza del formato dell'indirizzo ancora prima che i dati vengano inviati al server. Il tipo `password` si occupa di mascherare visivamente i caratteri digitati, `tel` è ottimizzato per l'inserimento di numeri di telefono — con il vantaggio, su dispositivi mobili, di richiamare una tastiera numerica dedicata — e `number` vincola l'input esclusivamente a valori numerici. Il tipo `submit`, infine, trasforma l'elemento in un pulsante che avvia l'invio del form, sebbene nella pratica contemporanea si preferisca spesso il tag `<button type="submit">`, che offre maggiore flessibilità dal punto di vista stilistico.

Quando si ha la necessità di raccogliere testo su più righe — come un messaggio o una descrizione estesa — si utilizza il tag `<textarea>`, che si distingue da `<input>` per avere tag di apertura e chiusura separati. Per proporre all'utente una selezione tra opzioni predefinite si ricorre invece a `<select>`, che genera un menu a tendina i cui elementi sono definiti tramite tag `<option>` annidati al suo interno. Le `<input type="radio">` assolvono invece a una funzione diversa: consentono scelte mutuamente esclusive all'interno di un gruppo, permettendo all'utente di selezionarne una sola alla volta.

Sul fronte dell'organizzazione del progetto, un principio architetturale di rilievo riguarda la separazione delle responsabilità tra struttura e presentazione visiva. I valori fondamentali di stile — colori di base e font tipografici — andrebbero centralizzati in un unico file `styles.css`, così da garantire coerenza visiva sull'intera applicazione e rendere le modifiche future sostanzialmente banali: cambiare il colore primario dell'interfaccia, per esempio, diventa un'operazione circoscritta a un solo punto del codice.

In merito alle librerie CSS, esistono soluzioni che mettono a disposizione classi predefinite pronte all'uso, evitando di scrivere CSS da zero. W3.CSS è una di queste: leggera e di semplice adozione, le sue classi si riconoscono per il prefisso `w3` e risulta adatta a prototipi o progetti di portata contenuta. Bootstrap rappresenta invece lo standard di riferimento in ambito professionale: più strutturata e potente, offre un sistema a griglia, una collezione completa di componenti UI e un ecosistema maturo che la rende adeguata a progetti di qualsiasi scala.

Riguardo alle scelte di design adottate nel progetto corrente, si è optato per un approccio flat, privo di ombre ed effetti di profondità, in favore di una semplicità visiva pulita e moderna. Il rosso è stato scelto come colore primario per le azioni di salvataggio, una scelta che andrà anch'essa centralizzata nel file `styles.css` globale, coerentemente con il principio di definire i valori fondamentali del design in un unico punto di riferimento.

In Angular, il collegamento tra la logica scritta in TypeScript e ciò che l'utente vede nel template HTML avviene attraverso un meccanismo chiamato data binding. Il componente TypeScript rappresenta lo stato dell'applicazione, mentre il template ne è la rappresentazione visiva: il data binding è il canale che mantiene i due livelli sincronizzati. La forma più elementare di questo collegamento è l'interpolazione, che si esprime con la doppia coppia di parentesi graffe `{{ }}` e permette di proiettare il valore di una variabile TypeScript direttamente come testo nel markup HTML. Si tratta di una comunicazione unidirezionale, dal componente verso il template.

`FormsModule` è il modulo di Angular che costruisce il ponte tra i campi di un form HTML e la logica del componente TypeScript. Per renderlo disponibile nell'applicazione è necessario importarlo esplicitamente.

`ngModel` è una direttiva Angular che implementa il two-way data binding su un campo di form, ovvero quel meccanismo per cui la variabile nel componente e il valore visualizzato nel campo HTML si mantengono costantemente allineati senza alcun intervento manuale. La sincronizzazione avviene in entrambe le direzioni in modo simultaneo: quando l'utente scrive nel campo, la proprietà del componente si aggiorna istantaneamente, e quando la proprietà muta per effetto della logica interna, il campo si aggiorna di conseguenza. Come regola pratica da tenere sempre presente: ogni volta che si utilizza `ngModel` all'interno di un tag `<form>`, l'attributo `name` sul campo è obbligatorio. Angular lo utilizza come identificatore per registrare quel controllo nell'ambito del form group complessivo, e la sua assenza rende il comportamento imprevedibile.

La sintassi "banana in a box" `[( )]` indica il binding bidirezionale: la scrittura nel campo aggiorna la variabile nel codice, e una modifica della variabile nel codice aggiorna il campo. Il property binding `[ ]` indica invece un binding unidirezionale, utilizzato quando si vuole visualizzare il risultato di una funzione — come `margin()` — senza consentire all'utente di modificarlo direttamente; in questo contesto viene tipicamente abbinato all'attributo `readonly`.

Vale la pena annotare che la modalità di lavoro descritta in questi appunti — basata su metodi di calcolo che vengono rieseguiti ogni volta che qualsiasi cosa cambia nello stato dell'applicazione — appartiene all'approccio tradizionale di Angular, antecedente all'introduzione dei signal e delle computed properties, e non è da considerarsi la pratica ottimale nel contesto attuale. Questa tecnica più vecchia è il change detection vecchio stile, ogni modifica di qualunque variabile, porta alla riesecuzione di tutti i metodi necessari al componente.