FrontEnd - Lezione del 09 febbraio 2026

Nello sviluppo del frontend, ogni entità del sistema deve essere rappresentata graficamente. In Java esiste una netta separazione tra entità e viste, poiché il linguaggio consente di associare molteplici rappresentazioni visive diverse a una singola entità. Nel contesto del frontend, invece, l'entità viene progettata specificamente in funzione della sua visualizzazione. Si inizia quindi a concepire la pagina come un insieme organizzato di componenti.

Il costruttore della classe Sanction è progettato per accettare come parametro un oggetto denominato state, il quale racchiude tutti i dati necessari relativi alla multa. All'interno del costruttore viene implementato un ciclo for-let che itera attraverso le proprietà presenti nell'oggetto state. Per ogni proprietà individuata, viene effettuata un'assegnazione dinamica al contesto corrente della classe utilizzando la notazione this[p], che imposta il valore corrispondente presente in state[p]. Questo approccio consente di popolare automaticamente l'istanza della classe con tutti i dati contenuti nell'oggetto state passato come argomento, garantendo una mappatura diretta e dinamica delle proprietà senza dover specificare manualmente ogni singolo campo.

La classe implementa inoltre un metodo denominato paid, il quale ha il compito di verificare lo stato di pagamento della multa. All'interno di questo metodo viene effettuato un controllo che restituisce il valore booleano risultante dal confronto tra la proprietà status dell'istanza corrente e la stringa "PAID". In questo modo è possibile creare un oggetto sanction a partire da un oggetto anonimo proveniente dall'esterno.

Per verificare il funzionamento dell'implementazione è stato creato un file test.html. Quando si lavora in JavaScript non è possibile sapere in anticipo quali proprietà possiedano gli oggetti, rendendo necessario questo approccio dinamico. Nel file test.html viene importato lo script sanction.js attraverso un tag script con attributo src. All'interno dello script di test viene definita una variabile state che contiene un oggetto letterale con le seguenti proprietà: id con valore 1, status con valore "PAID", ssn con valore "FIPPO", fee con valore "100", e cause con valore

"red light". Successivamente viene creata una nuova istanza della classe Sanction, assegnata alla variabile s, passando come argomento al costruttore l'oggetto state precedentemente definito. Vengono quindi eseguite due istruzioni console.log: la prima stampa l'intera istanza s per verificarne la struttura, mentre la seconda invoca il metodo paid sull'istanza s e ne stampa il risultato restituito.

Provando la pagina dal browser, il risultato restituito dal metodo paid è true, confermando che la multa risulta pagata. Questo approccio è necessario perché le API restituiscono oggetti anonimi: non arriva un'istanza completa di Sanction ma solamente lo stato della sanzione sotto forma di oggetto generico.

La classe implementa successivamente un metodo render che gestisce la visualizzazione dell'entità sanzione. Il metodo costruisce la struttura HTML utilizzando i backtick, che rappresentano una stringa speciale di JavaScript conosciuta come stringa di template. Il backtick consente di inserire variabili direttamente nella stringa attraverso l'interpolazione e permette di andare a capo liberamente secondo le necessità del codice. Per inserire il carattere backtick si utilizza la combinazione di tasti ALTGR più apostrofo.

All'interno del metodo render, la stringa di template contiene un elemento div con classe sanction e un attributo dinamico che determina lo stile in base allo stato di pagamento della multa. L'attributo verifica attraverso l'invocazione del metodo this.paid se la multa risulta pagata. Nel caso in cui il metodo restituisca true viene applicato lo stile green, mentre se restituisce false viene applicato lo stile red. Questo meccanismo consente di distinguere visivamente le multe pagate da quelle ancora da pagare.

All'interno del div principale si trova un secondo div che mostra l'identificativo della sanzione tramite l'espressione this.id seguito da un trattino e dalla motivazione della multa rappresentata da this.reason. Un ulteriore div visualizza la data della multa attraverso this.date seguito da un trattino, il simbolo dell'euro e l'importo della multa indicato da this.fee. Per inserire il simbolo dell'euro nel codice si tiene premuto il tasto alt e si digita 096 sul tastierino numerico.

Dopo aver completato la struttura HTML viene definito il foglio di stile CSS per formattare la presentazione della sanzione. Nel file test.html viene quindi invocato il metodo render per renderizzare effettivamente la sanzione nella pagina. A questo punto diventa fondamentale cominciare a ragionare sulla struttura

complessiva della pagina, considerandola come un insieme di componenti distinti che collaborano per formare l'interfaccia completa.

La classe Sanction è in grado di determinare autonomamente se una multa risulta pagata o meno attraverso il metodo paid. A questo punto dell'implementazione si procede con l'aggiunta di un'ulteriore entità: il cittadino (nel mock)

I dati relativi agli oggetti devono provenire dalle API, le quali inviano le informazioni che JavaScript si occuperà di rappresentare graficamente. Il ruolo di JavaScript è quello di gestire il rendering e la visualizzazione dei dati ricevuti. In questo paradigma di sviluppo frontend, un'entità non si limita esclusivamente a eseguire calcoli o operazioni logiche, ma è anche in grado di ragionare sulla propria struttura interna e di rappresentarsi graficamente attraverso metodi dedicati.

All'interno delle stringhe di template delimitante dai backtick, il simbolo del dollaro seguito da parentesi graffe permette di inserire espressioni JavaScript direttamente nella stringa. Questa sintassi si legge come interpolazione e consente di prendere un'espressione JavaScript, valutarla e incorporare il suo risultato all'interno della stringa. L'interpolazione rappresenta quindi il meccanismo attraverso il quale è possibile inserire dinamicamente il risultato di codice JavaScript all'interno di markup HTML, creando un ponte diretto tra la logica applicativa e la presentazione visiva.
Quando si affronta la necessità di renderizzare una collezione di elementi, come ad esempio un vettore di multe, è fondamentale comprendere l'architettura dei componenti autonomi e le loro responsabilità nel contesto del frontend moderno.

Componenti Autonomi e Responsabilità

Un componente frontend contemporaneo rappresenta molto più di un semplice contenitore di dati. Si configura piuttosto come un'entità intelligente caratterizzata da tre responsabilità fondamentali che ne definiscono il comportamento e la funzionalità. La prima responsabilità riguarda la gestione autonoma dei propri dati attraverso proprietà interne che vengono popolate in modo dinamico durante il ciclo di vita del componente. La seconda responsabilità si manifesta nell'implementazione di logica di business specifica, come avviene nel caso del metodo paid che ha il compito di determinare lo stato di pagamento della multa applicando regole e condizioni proprie del dominio applicativo. La terza responsabilità consiste nella capacità di auto-rappresentazione che il componente

esercita attraverso il metodo render, il quale genera autonomamente la struttura HTML necessaria alla visualizzazione del componente stesso all'interno dell'interfaccia utente.

Popolamento Dinamico delle Proprietà

Il costruttore della classe Sanction implementa un pattern di assegnazione dinamica delle proprietà attraverso un meccanismo basato su un ciclo che itera sull'oggetto state ricevuto come parametro di inizializzazione. Questo approccio progettuale si rivela necessario in JavaScript a causa della natura dinamica del linguaggio, che non prevede una tipizzazione statica delle classi come avviene in linguaggi fortemente tipizzati. Non è possibile conoscere in anticipo quali proprietà caratterizzeranno un oggetto proveniente da una API esterna, poiché la struttura dei dati può variare o evolversi nel tempo senza modificare il codice del componente. Il ciclo for-let esamina metodicamente ogni proprietà presente nell'oggetto state e la assegna all'istanza corrente utilizzando la notazione this[p], dove p rappresenta il nome della proprietà corrente. Questo meccanismo consente di mappare automaticamente qualsiasi struttura dati ricevuta dall'esterno senza dover definire manualmente ogni singolo campo nella classe, garantendo così flessibilità nell'integrazione con servizi esterni e riducendo significativamente la duplicazione del codice e la necessità di manutenzione.

Interpolazione e Template Literals

Le stringhe di template delimitate dai backtick rappresentano una funzionalità fondamentale di JavaScript moderno che permette di costruire markup HTML dinamico in modo elegante e leggibile. All'interno di queste stringhe è possibile inserire espressioni JavaScript utilizzando la sintassi dollar-parentesi graffe, meccanismo tecnicamente conosciuto come interpolazione. Quando il motore JavaScript incontra questa sintassi durante l'esecuzione del codice, valuta l'espressione contenuta tra le parentesi graffe e inserisce il risultato ottenuto nella stringa finale che verrà utilizzata per la renderizzazione. L'interpolazione consente di inserire variabili, chiamare metodi di istanza o statici e valutare espressioni condizionali direttamente nel contesto del markup HTML, creando così un collegamento diretto e reattivo tra lo stato interno dell'applicazione e la sua rappresentazione visiva nell'interfaccia utente.

Produzione dei Dati per il Componente

Produrre i dati necessari per alimentare un componente HTML rappresenta una sfida progettuale non banale che richiede una comprensione approfondita del data model sottostante. Il data model è composto da due entità correlate secondo una relazione padre-figlio. L'entità Citizen rappresenta il lato uno della relazione, fungendo da padre, ed è collegata in maniera bidirezionale all'entità Sanction che costituisce il lato molti della relazione. L'entità Citizen conterrà una lista di tipo List<Sanction> che raccoglie tutte le multe associate a quel particolare cittadino, mentre l'entità figlio Sanction conterrà un riferimento esplicito al padre attraverso una proprietà che punta all'oggetto Citizen di appartenenza.

Le API hanno la responsabilità di inviare al frontend esattamente i dati di cui ha bisogno per funzionare correttamente. Questo principio significa che ciò che arriva dal backend è stato progettato specificamente per soddisfare le necessità del client, tuttavia dal lato frontend rimane comunque necessario implementare controlli di validazione e sicurezza sui dati ricevuti. I dati che devono essere ricevuti includono l'identificativo della multa prelevato dall'entità Sanction, la data della multa anch'essa proveniente dall'oggetto di tipo Sanction, lo stato di pagamento contenuto in Sanction, la data di pagamento presente in Sanction, la motivazione della sanzione recuperata da Sanction e l'importo della multa sempre derivante da Sanction. A questi dati si aggiunge l'informazione relativa al cittadino che dovrà contenere nome e cognome del soggetto sanzionato, informazione che verrà calcolata a partire dall'oggetto Citizen, e per prudenza progettuale è opportuno includere anche l'identificativo del cittadino.

Il formato JSON utilizzato per la comunicazione non deve necessariamente corrispondere alla struttura degli oggetti presenti nel codice Java del backend. In Java non esiste una semplice stringa citizen ma un oggetto complesso di tipo Citizen con tutte le sue proprietà e relazioni. Le API sono progettate per parlare il linguaggio del client e inviano dati strutturati specificamente secondo le necessità e le convenzioni del frontend, operando una trasformazione dal modello di dominio interno del backend a un formato ottimizzato per il consumo da parte dell'interfaccia utente.