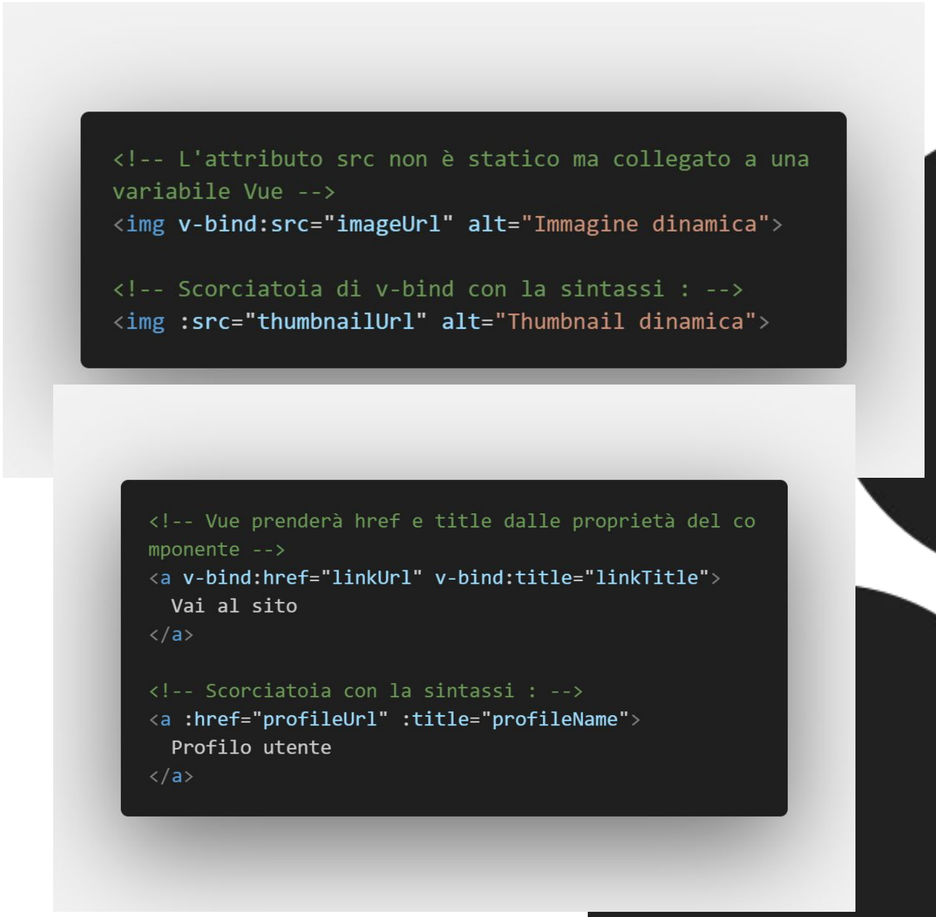
V-BIND

Lavorando al FE capita spesso di dover dinamicizzare attributi HTML, e per renderlo possibile si deve intervenire via JS.
Vue ha pensato anche a questo, infatti dispone di v-bind, una direttiva che serve proprio a “legare” (bind) i dati presenti in una variabile agli elementi del template.

Con v-bind i dati non solo vengono assegnati automaticamente, ma verranno anche aggiornati ogni volta che questi cambiano.

Lo schema di utilizzo è:
v-bind:attributo=”valore”
oppure
:attributo=”valore



V-BIND

V-bind è utile anche per l’assegnamento multiplo di attributi, per rendere questo possibile ti basta utilizzare un oggetto come valore.

```
<!-- "linkAttrs" potrebbe contenere { href: "...", ta
rget: "_blank", rel: "noopener" } -->
<a v-bind="linkAttrs">Apri link</a>
```

V-BIND

L’assegnamento di valori booleani richiede attenzione:

- Se il valore è true, Vue aggiunge l’attributo.
- Se è false o null/undefined, Vue lo rimuove.

$event

In js è possibile ottenere l’oggetto event leggendo il primo argomento ottenuto da una funzione legata ad un evento.
Talvolta però in vue la funzione legata ad un evento(ad esempio con v-on) riceve anche ulteriori argomenti personalizzati, questo rende l’oggetto Event non più raggiungibile.

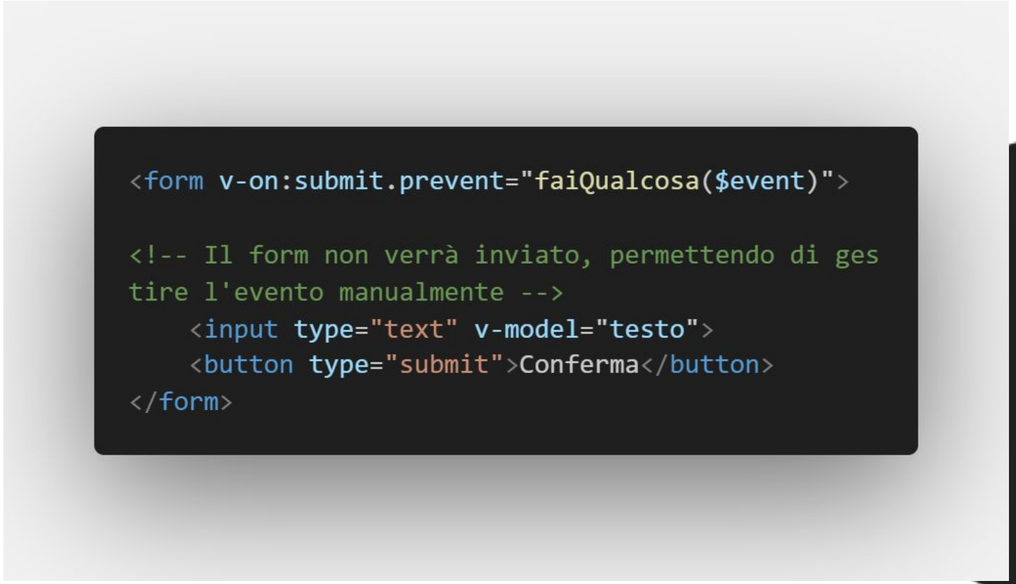
Ecco quindi che interviene $event, utilizzabile come argomento per richiamare in maniera esplicita le informazioni sul contesto.

```
<button v-on:click="faiQualcosa($event, nome)">
Cliccami</button>
```

Modificatori di evento

Quando si lavora con gli eventi nel framework, è spesso necessario introdurre logiche aggiuntive per controllarne il comportamento; i modificatori permettono di ottenere questo risultato direttamente nella dichiarazione dell’evento, senza dover scrivere ulteriore codice nel metodo associato.

Un esempio comune riguarda il modificatore .prevent, che richiama il metodo event.preventDefault() in maniera automatica. In questo modo, ad esempio, un form può intercettare l’evento di submit impedendo il ricaricamento della pagina. Analogamente, il modificatore .stop consente di bloccare la propagazione di un evento lungo la catena del DOM, evitando che un click su un elemento si rifletta anche sui suoi contenitori.



Modificatori di evento

Modificatore	Funzione	Esempio
`.stop`	Interrompe la propagazione dell'evento nel DOM (`event.stopPropagation()`).	`@click.stop="azione"`
`.prevent`	Impedisce il comportamento predefinito dell'evento (`event.preventDefault()`).	`@submit.prevent="invia"`
`.capture`	Usa la fase di cattura invece del bubbling.	`@click.capture="azione"`
`.self`	L'evento viene gestito solo se emesso dall'elemento stesso, non dai figli.	`@click.self="azione"`
`.once`	L'evento viene eseguito solo la prima volta che si verifica.	`@click.once="azione"`
`.passive`	Indica che l'evento non chiamerà `preventDefault()`, utile per performance (es. scroll).	`@scroll.passive="onScroll"`
`.enter`, `.tab`, `.delete`, `.esc`, `.space`, `.up`, `.down`, `.left`, `.right`	Vincolano l'evento a un tasto specifico della tastiera.	`@keyup.enter="conferma"`
`.ctrl`, `.alt`, `.shift`, `.meta`	L'evento viene gestito solo se il modificatore della tastiera è premuto.	`@click.ctrl="azione"`
`.left`, `.right`, `.middle`	Distinguono quale pulsante del mouse è stato premuto.	`@click.right="menu"`

Modificatori di evento

Si può anche gestire la pressione di tasti specifici, utilizzando la notazione con il codice del tasto o direttamente la lettera.

```
<!-- Quando l'utente preme "w" sulla tastiera -->
<input @keyup.w="muoviSu" placeholder=
"Premere W per muovere">
<!-- Keycode 87 corrisponde al tasto "W" -->
<input @keyup.87="muoviSu" placeholder=
"Premere W per muovere">

methods: {
    muoviSu() {
        alert("Hai premuto il tasto W!");
    }
}
```

v-if

La direttiva v-if consente un facile controllo condizionale della resa dei contenuti. Essa consente di stabilire se un determinato elemento debba o meno essere incluso nel DOM, in base al valore di un’espressione booleana o di un’espressione che possa essere valutata come tale.

A differenza di altre tecniche di semplice occultamento visivo, come l’uso della proprietà CSS display: none, l’utilizzo di v-if incide direttamente sulla struttura del DOM.

```
<p v-if="mostraMessaggio">
Questo testo sarà visibile solo se la condizione è
vera.
</p>
```

v-if-*

È importante osservare che Vue mette a disposizione anche le direttive complementari v-else-if e v-else, le quali consentono di costruire catene condizionali più articolate, analoghe alle istruzioni condizionali dei linguaggi di programmazione.

```
<p v-if="stato === 'successo'">
Operazione completata con successo.</p>
<p v-else-if="stato === 'errore'">
Si è verificato un errore.</p>
<p v-else>Stato non definito.</p>
```

v-for

Consente di iterare su una collezione di dati, come ad esempio un array o un oggetto, e generare automaticamente un insieme di elementi corrispondenti.

Immaginiamo di voler creare una lista di nomi partendo da un array. Un approccio statico, ogni elemento dovrebbe essere scritto manualmente nel codice HTML, rendendo l’operazione ridondante e poco scalabile.

Con v-for, invece, è sufficiente definire la struttura di un singolo elemento e demandare a Vue il compito di ripeterla per ogni voce della collezione.

```
<li v-for="(item, index) in items" :key="index">
  {{ index }} - {{ item }}
</li>
```

key serve a fornire a Vue un identificatore univoco per ciascun elemento generato.