



Vue.js

The Progressive JavaScript Framework

Vue.js

Vue è un **framework JavaScript** per la costruzione di interfacce utente e si basa su HTML, CSS e JavaScript. Fornisce un modello di programmazione **dichiarativo** e **basato su componenti** che aiuta a sviluppare interfacce utente di qualsiasi complessità.

Vue.js (esempio 0)

```
<button onclick="incrementCounter()" id="button">
  You clicked me: 0 times.
</button>

<script>
  let counter = 0;
  function incrementCounter() {
    counter++;
    const newText = `You clicked me: ${counter} times.`;
    document.getElementById('button').innerText = newText;
  }
</script>
```

Click counter - HTML + JS

Vue.js (esempio 0)

```
<script src="https://unpkg.com/vue@3"></script>

<div id="app">
  <button @click="counter++">
    You clicked me: {{ counter }} times.
  </button>
</div>

<script>
  Vue.createApp({
    data() {
      return { counter: 0 }
    }
  }).mount('#app')
</script>
```

1. Import del framework

2. Definizione del template

3. Definizione dello stato reattivo e installazione

Click counter - VUE

Vue.js

I concetti di **stato** e **template** permettono di introdurre due caratteristiche chiave di Vue:

- **Rendering dichiarativo:** tramite template, permette di descrivere in modo dichiarativo l'output HTML, in base allo *stato* JavaScript.
- **Reattività:** il DOM viene aggiornato quando si verificano modifiche allo *stato*.

Vue.js

```
<div id="app">
  <button @click="counter++">
    You clicked me: {{ counter }} times.
  </button>
</div>

<script>
  Vue.createApp({
    data() {
      return { counter: 0 }
    }
  }).mount('#app')
</script>
```

==

```
<div id="app"></div>

<script>
  Vue.createApp({
    data() {
      return { counter: 0 }
    },
    template: `
      <button @click="counter++">
        You clicked me: {{ counter }} times.
      </button>
    `
  }).mount('#app')
</script>
```

Nota: se il template non è specificato, Vue utilizza il contenuto HTML del contenitore su cui viene installato.

Vue.js

Vue è progettato per essere flessibile e adottabile in modo incrementale. A seconda del caso d'uso, può essere utilizzato in modi diversi:

- **HTML statico senza una fase di build**
- Tramite componenti in pagine web
- Single Page Application (SPA)
- Rendering lato server (SSR)
- Generazione di siti statici (SSG)
- Desktop, smartphone, WebGL e terminale

```
<script src="https://unpkg.com/vue@3"></script>
```

Direttive

Le **direttive** sono istruzioni speciali, prefissate con **v-** (e.g. `v-on`).

Permettono di aggiungere comportamento dinamico e reattivo agli elementi HTML, eseguendo operazioni che manipolano il DOM o ne modificano la visualizzazione. Si basano sullo stato o condizioni su esso.

Di seguito alcune direttive Vue:

- `v-bind:attributo="expr"`: associa un attributo HTML al risultato di `expr`.
- `v-model="variabile"`: binding bidirezionale tra input e variabile Vue.
- `v-if="cond"`: renderizza l'elemento solo se la condizione è vera.
- `v-else / v-else-if="cond"`: gestisce blocchi condizionali alternativi a `v-if`.

Direttive

- `v-for="item in items"`: cicla e crea un elemento per ciascun item.
- `v-on:evento="metodo"`: assegna un evento (e.g. `click`) ad un metodo Vue.
- `v-text="expr"`: inserisce solo testo all'interno di un elemento.
- `v-html="html"`: inserisce codice HTML nel DOM.
- `v-show="cond"`: controlla la visibilità dell'elemento con ``display: none``.
- `v-pre`: disattiva l'elaborazione Vue sull'elemento.
- `v-once`: renderizza il contenuto una sola volta e non lo aggiorna più.
- ...

Options API

Per la definizione di un componente possono essere utilizzate le **Options API**. Esse permettono di definire i dati reattivi e di specificarne la logica.

Di seguito alcune opzioni disponibili:

- `data()`: permette di definire i dati reattivi del componente.
- `methods`: permette di definire funzioni per la logica e interazioni del componente. Modificano lo stato e innescano aggiornamenti.
- `computed`: definizione di proprietà calcolate basate sui dati reattivi.

Options API

- `props`: definisce gli input ricevuti da un componente genitore.
- `watch`: permette di osservare i cambiamenti nei dati reattivi ed eseguire operazioni al verificarsi di essi.
- **Lifecycle hooks**: funzioni che vengono invocate durante il ciclo di vita del componente
 - `beforeCreate / created`
 - `beforeMount / mounted`
 - `beforeUpdate / updated`
 - `beforeUnmount / unmounted`
- ...

Options API (esempio 1)

```
Vue.createApp({
  data() {
    return {
      myReactiveState: 0
    }
  },
  computed: {
    myComputedState() {
      return "My reactive state is " + this.myReactiveState;
    }
  },
  methods: {
    increment() {
      this.myReactiveState++;
    }
  },
  mounted() {
    this.increment();
  }
}).mount('#app');
```

Esercizio 1

- Creare una pagina con Vue che consenta di visualizzare i film letti da server realizzato mediante Express.
- Partire dal codice messo a disposizione tra il materiale dell'esercitazione.

Esercizio 1

Obiettivo: visualizzare il **titolo**, **rating**, **plot**, **release date** (senza l'ora) e l'**immagine** dei film presenti nel file fornito (`src/controllers/movies.json`)

Amazing Movies

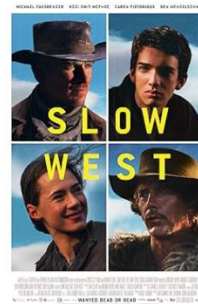


A Million Ways to Die in the West

Rating: R

As a cowardly farmer begins to fall for the mysterious new woman in town, he must put his new-found courage to the test when her husband, a notorious gun-slinger, announces his arrival.

Date: 2014-05-30



Slow West

Rating: R

A young Scottish man travels across America in pursuit of the woman he loves, attracting the attention of an outlaw who is willing to serve as a guide.

Date: 2015-04-16

Server

All'interno della cartella `exercise-01`, installare le dipendenze presenti nel file di configurazione ed eseguire il web server

```
npm install
```

```
node index.js
```

Suggerimento: le modifiche alla pagina html sono visibili dopo la ricarica del browser (non serve riavviare node)

Movies.html

- Nel file `www/movies.html` è già presente la struttura della pagina, inclusi:
 - Vue
 - Axios (per richieste HTTP, ma è possibile utilizzare anche altro, e.g. `Fetch API`)
 - Bootstrap
- Nel codice sono già presenti un container fluido e una card.
- Ogni film deve essere visualizzato in una card diversa.

Vue app

- Istanziare Vue, usando come id **movies-app**.
- Come dati, l'applicazione dovrà avere la lista di film, che inizialmente sarà vuota.

```
Vue.createApp({  
  data() {  
    return {  
      movies: []  
    }  
  }  
}).mount('#movies-app')
```

Vue app

- Aggiungere nei **methods** `listMovies` che dovrà effettuare una chiamata GET a `/api/movies`, leggere i dati e aggiungerli nella proprietà `movies`.

```
methods: {  
  listMovies(){  
    axios  
      .get("http://localhost:3000/api/movies")  
      .then(response => {  
        console.log(response.data)  
        // TODO  
      });  
  }  
}
```

More on Axios:

<https://axios-http.com/docs/intro>

Vue app

- `listMovies` deve essere chiamato all'avvio dell'applicazione
- Aggiungere la chiamata a `listMovies` all'interno di `mounted`
- Stesso livello di data e di methods

```
mounted() {  
    this.listMovies()  
}
```

Vue app

- Dopo aver letto i dati, visualizzarli nell'html.
- Nel div con classe card, aggiungere un ciclo sui film (`v-for`)

```
<div class="card" v-for="movie in movies" :key="movie.id">
```

Vue app

- Nel div con classe col-md-4, inserire l'immagine

```

```

Vue app

Alcune immagini non sono più disponibili, sostituirle con una di default

- Nell'immagine:

`@error="replaceByDefault"`

- Tra i metodi:

```
replaceByDefault(event) {  
    event.target.src = defaultImgUrl  
}
```

Vue app

- Nel div con classe card-body, inserire gli altri dati (si può usare interpolazione del testo `{{ }}` in alternativa)

```
<h5 class="card-title" v-text="movie.title"></h5>
<p class="card-text" v-text="movie.rated"></p>
<p class="card-text" v-text="movie.plot"></p>
<p class="card-text" v-text="movie.released ? movie.released.substring(0,10)
: 'N/A'"></p>
```

Vue app

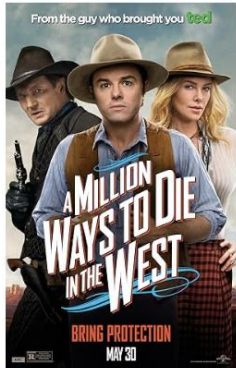

- Modificare l'URL di tutte le immagini appena ricevuta la risposta dal server
 - <http://ia.media-imdb.com/> → <https://m.media-amazon.com/>

Esercizio 2

Obiettivo: creare una pagina con Vue che consenta di visualizzare i film, aggiungerne di nuovi tramite form e cancellarli dall'elenco, basandosi sulla struttura fornita.

CRUD Movies

+Add Movie

Title	Plot	Poster	Release date	Actions
A Million Ways to Die in the West	As a cowardly farmer begins to fall for the mysterious new woman in town, he must put his new-found courage to the test when her husband, a notorious gun-slinger, announces his arrival.		2014-05-30T04:00:00Z	

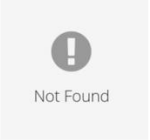





Esercizio 2

- Inserimento dati
 - Quando l'utente clicca su **Add Movie**, deve essere mostrato il form, altrimenti è nascosto
 - Il click su **Cancel** deve nascondere il form, svuotandolo.
 - Il click su **Submit** deve inserire i dati.

Esercizio 2

CRUD Movies

+Add Movie

Title	Plot	Poster	Release date	Actions
Once Upon a Time in the West	Epic story of a mysterious stranger with a harmonica who joins forces with a notorious desperado to protect a beautiful widow from a ruthless assassin working for the railroad.			
A Million Ways to Die in the West	As a cowardly farmer begins to fall for the mysterious new woman in town, he must put his new-found courage to the test when her husband, a notorious gun-slinger, announces his arrival.		2014-05-30T04:00:00Z	
Wild Wild West	The two best hired guns in the West must save President Grant from the clutches of a nineteenth-century		1999-06-30T04:00:00Z	



CRUD Movies

+Add Movie

Title

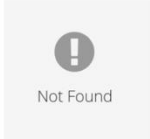

Plot

Poster link

Release date

Submit

Cancel

Title	Plot	Poster	Release date	Actions
Once Upon a Time in the West	Epic story of a mysterious stranger with a harmonica who joins forces with a notorious desperado to protect a beautiful widow from a			

Esercizio 2

- La struttura di partenza è nel file `www/movies-crud.html` accessibile tramite la rotta `/movies-crud`
- Consigliato l'utilizzo di Axios per le richieste HTTP
- Utilizzare l'istanza di Vue creata precedentemente, modificando l'id di installazione del componente da `movies-app` a **`crud-movies-app`**.

Vue app

- Inserire i dati dei film nel template della tabella
 - Aggiungere una riga (<tr>) per ogni film (dentro <tbody>):

```
<tr v-for="movie in movies" :key="movie.id">
  <td>{{ movie.title }}</td>
  <td>{{ movie.plot }}</td>
  <td></td>
  <td>{{ movie.released }}</td>
  <td>
    <button @click.prevent="deleteMovie(movie.id)" ...>
      <i class="fas fa-trash-alt"></i>
    </button>
  </td>
</tr>
```

Vue app

- Nascondere il form
 - Aggiungere il seguente codice al contenitore del form di inserimento
`<div class="row" v-if="adding">`
 - Aggiungere la proprietà `adding`, inizializzandola a `false`

```
data() {  
  return {  
    movies: [],  
    adding: false  
  }  
}
```

Vue app

- Per mostrare il form aggiungere un listener per il click sul bottone

@click="showAddMovieForm"

- Definire il metodo `showAddMovieForm`

```
showAddMovieForm() {  
    this.adding = true;  
}
```

Vue app

- Per nascondere il form aggiungere un listener per il click sul bottone.

@click.prevent="hideAddMovieForm"

- Definire il metodo `hideAddMovieForm`

```
hideAddMovieForm() {  
    this.adding = false;  
}
```


Vue app

- Definire una proprietà `newMovie`, che conterrà i dati inseriti dall'utente.

```
data() {  
  return {  
    movies: [],  
    adding: false,  
    newMovie: { }  
  }  
}
```

Vue app

- Creare un metodo di `resetNewMovie` che permette di eliminare il contenuto del form

```
resetNewMovie() {  
  this.newMovie = {  
    "title" : "",  
    "plot" : "",  
    "poster" : "",  
    "released" : new Date().toISOString().slice(0,10)  
  }  
}
```

Vue app

- Chiamare il metodo `resetNewMovie` all'interno di `hideAddMovieForm`

```
hideAddMovieForm() {  
    this.adding = false;  
    this.resetNewMovie();  
}
```

Vue app

- Collegare la proprietà **newMovie** agli elementi del form
 - È sufficiente aggiungere la proprietà `v-model` ad ogni input
 - Visualizzare l'immagine

`v-model="newMovie.title"`

`v-model="newMovie.plot"`

`v-model="newMovie.poster"`

`v-bind:src="newMovie.poster"`

`v-model="newMovie.released"`

Vue app

- Definire un metodo per inserire **newMovie**.

```
addMovie() {  
  axios.post('/api/movies', this.newMovie)  
    .then(response => {  
      this.movies.push(response.data);  
      this.hideAddMovieForm();  
    });  
}
```

Vue app

- Specificare onclick del bottone **Submit**
`@click.prevent="addMovie"`

Vue app

- Definire il metodo per la cancellazione

```
deleteMovie (movieId) {  
  axios.delete (/api/movies/' + movieId)  
    .then (response => {  
      this.movies.splice (  
        this.movies.findIndex (item => item.id === movieId), 1)  
      });  
    }  
}
```

Vue app

- Specificare onclick del bottone **Trash**
`@click="deleteMovie(movie.id)"`