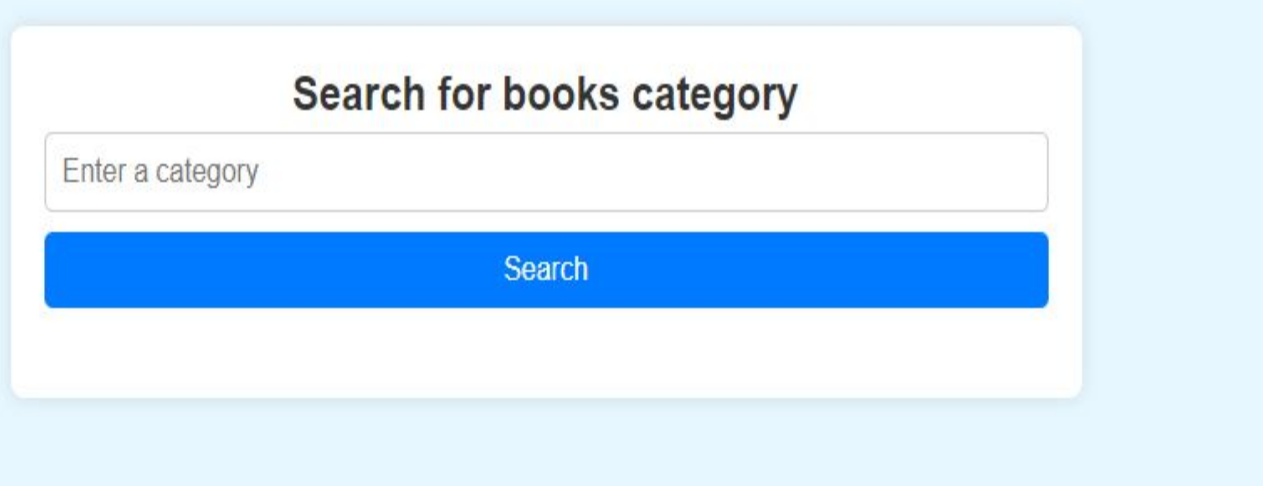


Presentazione progetto JavaScript Advanced

Di Ricci Matteo

L'app BookSearch

L'applicazione “BookSearch” presentata di seguito è un'app che permette all'utente di cercare una categoria di libri all'interno di un text-box. Dopo aver scelto ed inserito una categoria, l'applicazione mostra i libri che vi appartengono. L'utente può quindi scegliere di cliccare sull'apposito tasto “Show description” per visualizzare una breve descrizione del libro interessato.



The image shows a screenshot of a web application interface for searching book categories. It features a light blue background with a subtle wave pattern. The main content is a white rounded rectangle with a thin grey border. Inside this rectangle, the title "Search for books category" is centered at the top in a bold, dark grey font. Below the title is a text input field with a light grey border and the placeholder text "Enter a category" in a light grey font. At the bottom of the white rectangle is a solid blue button with the word "Search" written in white, centered text.

File Index.html

Innanzitutto ho creato il file “index.html”, dove è presente un container che racchiude un input-box che permette la ricerca della categoria del libro, e un div dove saranno visualizzati i risultati una volta premuto il pulsante.

```
<> index.html > html
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Ricerca Libri</title>
7
8      <link rel="stylesheet" href="/css/styles.css">
9  </head>
10 <body>
11     <div class="container">
12         <h1>Search for books category</h1>
13         <input type="text" id="category" placeholder="Enter a category">
14         <button id="searchButton">Search</button>
15         <div id="results"></div>
16     </div>
17
18     <script src="/js/script.js"></script>
19 </body>
20 </html>
```

Il codice Javascript

Prima di tutto ho creato una funzione “onclick” del pulsante: una volta inserita la categoria, se questa esiste, l’app esegue la funzione “fetchbooks”, altrimenti si riceverà un messaggio di errore.

```
js > JS script.js > ...
1 document.getElementById('searchButton').addEventListener('click', function () {
2     const category = document.getElementById('category').value;
3     if (category) {
4         fetchBooks(category);
5     } else {
6         alert('Please, insert a valid category.');
```


A questo punto ho creato la funzione “fetchbooks” con cui, facendo riferimento all’API “openlibrary” fornita nella descrizione del progetto, l’app fornisce un elenco di tutti i libri presenti nella categoria. Per fare ciò ho usato la funzione “fetch”, che cerca nell’url con la categoria inserita dall’utente e poi mostra i risultati grazie alla funzione “displayResults”.

```
async function fetchBooks(category) {  
  const apiUrl = `https://openlibrary.org/subjects/${category}.json`;  
  const headers = new Headers({  
    "User-Agent": "librarySearch/1.0 (matteo.ricci0304@gmail.com)"  
  });  
  
  try {  
    const response = await fetch(apiUrl, {  
      method: 'GET',  
      headers: headers  
    });  
  
    if (!response.ok) {  
      throw new Error('Network response was not ok');  
    }  
  
    const data = await response.json();  
    const books = _.get(data, 'works', []);  
    displayResults(books);  
  } catch (error) {  
    console.error('Fetch error:', error);  
    alert('An error occurred while fetching book data. Please try again later.');
```

La funzione “displayResults” fa riferimento al div creato nel file “index.html”. Nella prima parte, nel caso in cui la categoria cercata dall’utente non esista o non includa alcun libro appartenente ad essa, verrà mostrato un messaggio relativo al fatto che non sono stati trovati libri.

```
function displayResults(books) {  
  const resultsDiv = document.getElementById('results');  
  resultsDiv.innerHTML = '';  
  
  if (books.length === 0) {  
    resultsDiv.innerHTML = '<p>No book has been found for this category.</p>';  
    return;  
  }  
}
```

A questo punto, cosa succede se la categoria cercata dall'utente è valida? Prima di tutto per ogni libro appartenente ad essa, viene cercato il valore "title" che contiene appunto il titolo del libro. Viene poi creato un "h3" che mostra il titolo. La costante titleElement è stata poi "appesa" alla costante-genitore bookElement.

Per gli autori, ho creato una costante "authorElement", che cerca il valore "name" all'interno di "authors" e lo mostra con una riga di testo. Ho proceduto poi ad attaccare anche questa costante a "bookElement".

```
book.authors.forEach(author => {  
  const authorElement = document.createElement('p');  
  authorElement.textContent = `Author: ${author.name}`;  
  bookElement.appendChild(authorElement);  
  authorElement.appendChild(descriptionButton);  
  resultsDiv.appendChild(bookElement);  
})
```

```
books.forEach(book => {  
  const bookElement = document.createElement('div');  
  bookElement.className = 'book';  
  
  const titleElement = document.createElement('h3');  
  titleElement.textContent = book.title;  
  bookElement.appendChild(titleElement);
```

```
{  
  "key": "/subjects/love",  
  "name": "love",  
  "subject_type": "subject",  
  "work_count": 4918,  
  "works": [  
    {  
      "key": "/works/OL66534W",  
      "title": "Pride and prejudice",  
      "edition_count": 752,  
      "authors": [  
        {  
          "name": "Jane Austen",  
          "key": "/authors/OL21594A"  
        }  
      ],  
      "has_fulltext": true,  
      "ia": "mansfieldparknov03aust",  
      ...  
    },  
    ...  
  ]  
}
```


Entriamo nella sezione riguardante la descrizione del libro. Innanzitutto ho creato un pulsante che, una volta cliccato, mostra un breve sommario del libro in questione (funzione “toggleDescription”)

```
const descriptionButton = document.createElement('button');
descriptionButton.textContent = 'Show Description';
descriptionButton.addEventListener('click', function () {
  toggleDescription(book.key, bookElement, descriptionButton);
});
```

Per farlo, ho pensato che come prima cosa sarebbe stato corretto implementare un funzionamento col quale, una volta mostrata la descrizione, fosse anche possibile cliccare di nuovo sul pulsante e farla sparire. Il testo del pulsante cambia quindi da “Show Description” a “Hide Description” quando la descrizione viene mostrata all’utente.

```
function toggleDescription(bookKey, bookElement, button) {
  let descriptionElement = bookElement.querySelector('.description');
  if (descriptionElement) {
    if (descriptionElement.style.display === 'none') {
      descriptionElement.style.display = 'block';
      button.textContent = 'Hide Description';
    } else {
      descriptionElement.style.display = 'none';
      button.textContent = 'Show Description';
    }
  } else {
    fetchDescription(bookKey, bookElement, button);
  }
}
```


In ultimo, ho creato la funzione “fetchDescription”. Facendo riferimento ad una diversa API che contiene la descrizione dei libri, identificati tramite key, la funzione passa la key corrispondente al libro. Utilizzando la libreria Lodash e la sua funzione “_.get”, viene cercato il valore “data”, ma non essendo sicuro che questo esista il terzo valore di default è stato impostato come “No description available”. In questo caso, se il valore “data” di un libro non esiste, il valore mostrato dall’app sarà appunto il messaggio “No description available”.

```
async function fetchDescription(bookKey, bookElement, button) {
  const apiUrl = `https://openlibrary.org${bookKey}.json`;
  try {
    const response = await fetch(apiUrl);

    if (!response.ok) {
      throw new Error('Network response was not ok');
    }

    const data = await response.json();
    const description = _.get(data, 'description.value', 'No description available.');
```

```
const descriptionElement = document.createElement('p');
descriptionElement.className = 'description';
descriptionElement.textContent = description;
bookElement.appendChild(descriptionElement);
button.textContent = 'Hide Description';
  } catch (error) {
    console.error('Fetch description error:', error);
    alert('An error occurred while fetching the book description. Please try again later.');
```

```
  }
}
```

Applicando poi del CSS, il risultato è l'app visualizzata nell'immagine.

L'app è stata quindi deployata su Netlify ed è visualizzabile al seguente link:

<https://booklibrarysearch.netlify.app>.

Ho quindi creato il file README.md, visualizzabile insieme all'intero codice.

NOTA: dato il limite massimo di 10Mb caricabili sul sito, ho deciso di non uploadare la cartella webpack.

L'intero codice con tanto di cartella webpack è visualizzabile al mio [Github](#).

E questo conclude il mio progetto sull'app "BookSearch". Vi ringrazio anticipatamente per il feedback e per aver visualizzato la mia presentazione!

Ricci Matteo

Search for books category

Search

Alice's Adventures in Wonderland

Author: Lewis Carroll

Hide Description

Alice's Adventures in Wonderland (commonly Alice in Wonderland) is an 1865 English children's novel by Lewis Carroll. A young girl named Alice falls through a rabbit hole into a fantasy world of anthropomorphic creatures. It is seen as an example of the literary nonsense genre. One of the best-known works of Victorian literature, its narrative, structure, characters and imagery have had huge influence on popular culture and literature, especially in the fantasy genre.

Treasure Island

Author: Robert Louis Stevenson

Show Description

Gulliver's Travels

Author: Jonathan Swift

Show Description

Through the Looking-Glass

Author: Lewis Carroll

Show Description