

Esercitazione PHP e AJAX



Outline

- Esercizi Php e Ajax



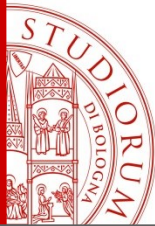
Materiale a disposizione

- Il materiale dell'esercitazione contiene una cartella con il codice dello scorso laboratorio (**con qualche aggiunta**). Nella cartella db trovate sempre gli script per creare e popolare il db.



Esercizi

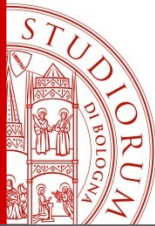
- Nel laboratorio precedente abbiamo creato tutte le pagine del sito.
- Oggi vedremo una versione alternativa delle pagine, usando Ajax. Il server restituirà le pagine con il tag main vuoto. Al caricamento della pagina, verrà effettuata una richiesta http per recuperare i dati dal server.
- Per le richieste Ajax, utilizzeremo l'istruzione fetch



fetch

- fetch è un'API nativa di JavaScript per fare richieste HTTP e recuperare risorse, come dati da un server.
- Caratteristiche principali:
 - Basato su Promise
 - Gestione delle risposte tramite `.then()` e `.catch()`.
 - Moderno e flessibile rispetto a `XMLHttpRequest`.

```
fetch('https://api.example.com/data')  
  .then(response => response.json())  
  .then(data => console.log(data))  
  .catch(error => console.error('Errore:', error));
```



Promise

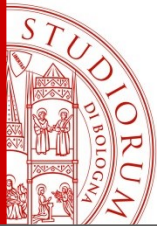
- Una Promise rappresenta un'operazione asincrona che può essere:
 - Risolta con un valore.
 - Rifiutata con un errore.
- Possibili stati:
 - Pending
 - Fulfilled
 - Rejected

```
const promise = new Promise((resolve, reject) => {  
  if (success) resolve('Successo!');  
  else reject('Errore!');  
});  
  
promise  
  .then(value => console.log(value))  
  .catch(error => console.error(error));
```



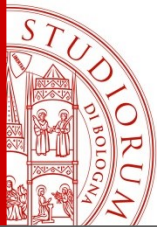
Promise - Esempio

```
function fintaOperazione() {  
  return new Promise((resolve, reject) => {  
    setTimeout(() => {  
      const successo = Math.random() > 0.5;  
      if (successo) resolve('Operazione completata!');  
      else reject('Operazione fallita.');    }, 2000);  
  });  
}  
  
fintaOperazione()  
  .then(result => console.log(result))  
  .catch(error => console.error(error));
```



async e await

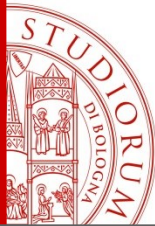
- `async` e `await` sono due costrutti introdotti per semplificare la gestione del codice asincrono. Permettono di scrivere codice asincrono come se fosse sincrono.
- `async`:
 - Utilizzata per dichiarare una funzione asincrona.
 - Una funzione `async` restituisce sempre una `Promise`.
- `await`:
 - Utilizzabile solo all'interno di una funzione `async`.
 - Sospende l'esecuzione fino al completamento della `Promise`.



Esempio

```
fetch('https://api.example.com/data')  
  .then(response => response.json())  
  .then(data => console.log(data))  
  .catch(error => console.error('Errore:', error));
```

```
async function getData() {  
  try {  
    const response = await fetch('https://api.example.com/data');  
    const data = await response.json();  
    console.log(data);  
  } catch (error) {  
    console.error('Errore:', error);  
  }  
}  
  
getData();
```



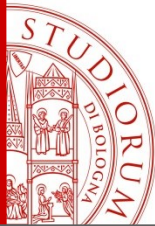
Confronto Promise e async/await

- Promise:
 - Maggiore flessibilità
 - Adatta per pipeline complesse con `.then()` concatenati.
 - Leggibilità del codice annidato è ridotta.
- `async/await`:
 - Più leggibile per operazioni sequenziali.
 - Riduce il rischio di "callback hell".
 - Richiede la gestione esplicita degli errori con `try/catch`.



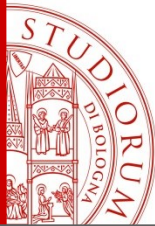
fetch - Opzioni

```
const myHeaders = new Headers();  
myHeaders.append("Content-Type", "application/json");  
  
const response = await fetch("https://example.org/post", {  
  method: "POST",  
  body: JSON.stringify({ username: "example" }),  
  headers: myHeaders,  
});
```



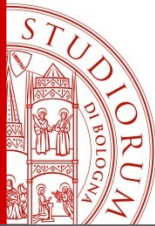
fetch - Opzioni

- Passare i parametri nel body come stringify non porterà i parametri ad essere inseriti nell'array `$_POST`.
- Per fare sì che ciò avvenga, bisogna:
 - Creare un oggetto di tipo `FormData`
`const formData = new FormData();`
 - Inserire al suo interno i parametri con il metodo `append`:
`formData.append('nomecampo', valorecampo);`
- Passare l'oggetto `formData` nel corpo della richiesta:
...
`body: formData`
...



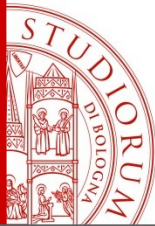
Esercizio 01

- Popolare la homepage (creare un nuovo file `index2.php`) utilizzando Javascript, richiedendo i dati con Ajax:
 - `api-articolo.php` dovrà restituire un json contenente gli ultimi due articoli inseriti sul db.
 - `index2.php` non avrà un template specifico, ma restituirà il tag `main` vuoto.
 - `index.js` dovrà effettuare una richiesta `http get` alla pagina `api-articolo.php` e creare la struttura `html` utilizzando Javascript.



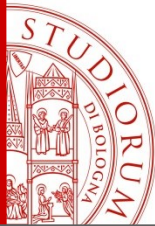
Esercizio 02

- Popolare la pagina contatti (creare un nuovo file contatti2.php) utilizzando Javascript, richiedendo i dati con Ajax:
 - api-contatti.php dovrà restituire un json contenente i dati degli autori presenti sul db.
 - contatti2.php non avrà un template specifico, ma restituirà il tag main vuoto.
 - contatti.js dovrà effettuare una richiesta http get alla pagina api-contatti.php e creare la struttura html utilizzando Javascript.



Esercizio 03

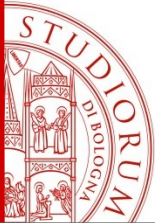
- Gestire la pagina login (creare un nuovo file login2.php) utilizzando Javascript, richiedendo i dati con Ajax:
 - api-login.php dovrà:
 - in caso di richiesta get, restituire un oggetto json con proprietà logineseguito che vale true se l'utente è loggato, false altrimenti.
 - in caso di richiesta post, eseguire login, restituendo eventuali errori nel campo errorelogin.
 - in generale, se l'utente è loggato restituire nel campo articolautore gli articoli scritti dall'utente.
 - login2.php non avrà un template specifico, ma restituirà il tag main vuoto.



Esercizio 03

– login.js dovrà:

- Al caricamento della pagina fare una richiesta get alla pagina api-login.
- Se il login è stato eseguito, visualizzare gli articoli.
- Altrimenti, visualizzare il form per il login e gestire il submit del form con Javascript. La richiesta Post deve essere effettuata con fetch (attenzione a come inviare i parametri in modo che siano leggibili da Php).
- In caso di errore, visualizzare il messaggio nel paragrafo.
- Altrimenti visualizzare gli articoli.



Esercizio A

- Modificare il sito in modo che ci sia un'unica pagina index.php e la gestione dei contenuti del main sia gestita interamente con Javascript.
- Gestire solo le seguenti pagine:
 - Home
 - Contatti
 - Login



Esercizio B

- Modificare il sito in modo che ci sia un'unica pagina index.php e la gestione dei contenuti del main sia gestita interamente con PHP.
- Gestire solo le seguenti pagine:
 - Home
 - Contatti
 - Login