JavaScript: upload di file con le Fetch API

In JavaScript le Fetch API ci consentono di effettuare anche l'upload di file.

Si tratta di utilizzare l'oggetto `FormData` inserendolo come contenuto della proprietà `body` di una richiesta POST:

```
'use strict';

const uploadFile = async ({ url, fileName, fileData
}) => {
    const data = new FormData();
    data.append(fileName, fileData);

    try {
        const req = await fetch(url, { method:
'POST', body: data });
        return await req.json();
    } catch(err) {
        return null;
    }
};
```

La funzione `uploadFile()` accetta come parametro un oggetto con tre proprietà:

1. `url`: l'endpoint dell'applicazione che gestirà l'upload del file;

2. `fileName`: il nome del file così come appare come attributo `name` di un campo di input `file`;
3. `fileData`: un riferimento all'oggetto file ottenuto accedendo all'array `files` del campo di input.

Per evitare le funzioni di callback create da `then()`, la funzione utilizza il modello async/await per gestire le Promise create dalla funzione `fetch()`. Restituisce un oggetto JSON in caso di avvenuto upload o `null` in caso di errore.

Possiamo usarla con un form di upload nel modo seguente:

```
const form = document.getElementById('upload');
const fileInput = form.querySelector('#file');

form.addEventListener('submit', evt => {
    evt.preventDefault();
    const params = {
        url: '/api/upload',
        fileName: 'file',
        fileData: fileInput.files[0]
    };
    uploadFile(params).then(results => {
        //...
    });
}, false);
```

Intercettiamo l'evento `submit` del form impedendo che quest'ultimo venga inviato in modalità sincrona. Quindi usiamo la nostra funzione specificando l'oggetto che contiene i parametri per effettuare l'upload.