Übung 13: Fetch

Aufgabe: Fetch im Server

In der vorherigen Übung haben Sie einen RESTful Service für Ihren Blog gebaut und diesen mittels Postman oder ThunderClient getestet. In der SU haben Sie zudem asynchrone Requests über „fetch“ kennengelernt. Nutzen Sie Ihr gewonnenes Wissen, um Ihren Blog (blog.js-Route) derart zu verändern, dass diese Ihren RESTful Service nutzt.

Beginnen Sie am besten mit der Anzeige aller Posts und eines einzelnen Posts.

Route	Methode		Anzeige/View
„/“	GET	Rufen Sie über node-fetch Ihre eigene API auf, welche Ihnen alle Blogeinträge liefert. Nutzen Sie den „response“ dieses fetch-Aufrufs, um Ihre „blog“-View mit den entsprechenden Daten zu füllen.	Alle Blogeinträge in „Cards“ View: blog.ejs
„/:postID“	GET	Rufen Sie über node-fetch Ihre eigene API auf, welche Ihnen den entsprechenden Blogeintrag liefert. Nutzen Sie den „response“ dieses fetch-Aufrufs, um Ihre „post“-View mit den entsprechenden Daten zu füllen.	Detail eines Posts

Blog.js – Datei (Routes)

1. Schreiben Sie eine asynchrone Funktion „handleAPICalls“, welche die Variablen „url“, „method“ und „body“ entgegennehmen kann.
2. Definieren Sie innerhalb dieser Funktion eine Variable „response“.
3. Erstellen Sie innerhalb dieser Funktion eine fetch-Anfrage (wie in der Vorlesung gezeigt), welche die von Ihnen definierten Variablen „url“, „method“ und „body“ an entsprechender Stelle nutzt und warten auf die Antwort des Servers. Diese Antwort speichern Sie in der „response“ Variable geben diese als return zurück.
   ```
   response = await fetch(…)
   ```
4. Als letzten Schritt müssen Sie die Funktion „handleAPICalls“ in den oben beschriebenen Routen Ihre eigene API anzufragen und die jeweiligen „return’s“ an Ihre View (res.render) übergeben.

Aufgabe: Kombinierte Views

Sehen Sie sich folgenden Screenshot der Datei „post.ejs“ einmal in Ruhe an.

```
<%- include('particals/header') %>
    <div id="formular">
        <%- include('particals/formular') %>
    </div>
    <div id="viewpost">
        <% if(typeof(post)!='undefined' ) { %>
            <%- include('particals/viewpost') %>
        <% } %>
    </div>

    <!-- Hier kommt Fetch (post, put und delete) sowie -->
    <!-- das umschalten zwischen formular und viewpost -->
    <script>⋯
    </script>

    <%- include('particals/footer') %>
```

Wie Sie sehen, includiert die Datei die „Particals“: header, formular, viewpost und footer. Alle diese Elemente haben Sie schon in Ihren Views als einzelne EJS-Dateien erstellt.

Header: Beinhaltet den Header, sowie die Navigation.
Formular: Beinhaltet ein Formular, welches dynamisch erstellt werden kann.
Viewpost: Beinhaltet ein einzelnes „Card“-Element, welches einen Post anzeigt.
Footer: Beinhaltet den Footer

1. Nutzen Sie Ihre „Particals“, um eine solche View (post.ejs) zu erstellen. Falls Sie eine der hier beschriebenen Views noch nicht haben, sollten Sie dies nachholen.
2. Lassen Sie diese View (post.ejs) von der Route „/:postID“ anzeigen (res.render….)
3. Das Formular soll die gleichen Felder beinhalten, welche für das Erstellen eines neuen Posts notwendig sind.
4. Nutzen Sie das Value-Attribut, um die schon vorhandenen Werte dieses Posts anzuzeigen.
   WICHTIG: Neben den „normalen“ Einträgen müssen Sie auch die ID des jeweiligen Posts in das Formular übergeben. Nutzen Sie hierfür den Typ „hidden“.
   `{ label: "", type: "hidden", name: "id", required: true, value: result.data.id},`
5. Schreiben Sie innerhalb des „script“-Tags einen js-Code, welcher das DIV-Element mit der id „formular“ ausblendet.
6. Implementieren Sie in der View „viewpost.ejs“ einen Link zum „Editieren“ des Posts
   `<a href=“#“ id=“editPost“>Editieren</a>`
7. Erstellen Sie im script-Tag einen Eventlistener für den gerade implementierten Button. Dieser Eventhandler soll beim Klick auf den Button das div mit der id „formular“ wieder anzeigen und gleichzeitig das div mit der id „viewpost“ ausblenden.

Hinweis: Mit diesen Vorbereitungen können Sie später (nachdem Sie die nächste Aufgabe abgeschlossen haben) auch einen neuen Post mit dieser View an den Server senden und somit in der Route „newPost“ diese View aufrufen.

Aufgabe: Fetch im Client

Bisher haben Sie „Fetch im Server“ und die „Kombinierte View“ fertiggestellt. Nun sollen Sie die http-Methoden PUT, POST und DELETE in der EJS-Datei (post.ejs) implementieren.
WICHTIG: Diese Anleitung geht davon aus, dass Sie „express-fileupload“ installiert haben!

1. Beginnen Sie, indem Sie eine Funktion „apiCall“ in Ihrem script-Tag aus der vorherigen Übung erstellen. Diese Funktion nimmt die Variablen „method“ und „id=‘‘ “ entgegen. Der Funktionskopf sieht also wie folgt aus: `function apiCall(method, id=““) { … }`
2. In dieser Funktion erstellen Sie nun den fetchaufruf zu Ihrem Server. Die Funktion „apiCall“ bzw. deren Inhalt sollen erst durchgeführt werden, wenn in Ihrem Formular der „submit“-Button gedrückt wird. Also erstellen Sie für diesen Button einen weiteren Eventlistener und müssen innerhalb von diesem als erstes das „default“-Event unterdrücken.
3. Danach müssen Sie die eingegebenen Daten aus dem Formular einlesen und in einer Variable speichern. Eine Möglichkeit dies zu tun ist die „FormData“-Klasse, welcher das DOM-Element des Formulars übergeben wird. Auch hier erhalten Sie ein Beispiel:
```
let data = new FormData(document.getElementById('meinFormular'));
data.append('files', document.querySelector('input[type="file"]').files[0]);
```
4. Jetzt haben Sie alle Bestandteile des fetchaufrufs zusammen und können diesen durchführen.
5. Wenn Ihnen der Server eine Antwort sendet, sollen Sie eine Weiterleitung auf den entsprechenden Blogeintrag implementieren. Der „Inhalt“ der .then-Funktion wäre:
`window.location.href = "http://localhost/blog/"+ data.data.id;`
Wobei “data.data.id” in diesem Fall die ID des jeweiligen Blogeintrags darstellt.
6. Sie haben nun die Funktion „apiCall“ erfolgreich implementiert. Jetzt müssen Sie diese nur noch, je nachdem welcher Link gedrückt wird, aufrufen.

Funktion	URL	Methode	Body
Neu Erstellen	http://localhost/blog	POST	Formular
Editieren	http://localhost/blog/id	PUT	Formular

7. Eine Besonderheit stellt “delete” dar, da hier nur der Button gedrückt wird und kein „submit“ eines Formulars. Daher müssen Sie hier einen separaten fetchaufruf implementieren.

*Aufgabe: Server-To-Server-Fetch**

Nutzen Sie die API: https://api.chucknorris.io innerhalb Ihres Servers (blog.js) und speichern Sie beim Erstellen eines neuen Posts zusätzlich einen Witz.