Webentwicklung (WE) - FWPM

Übung 9: “Javascript im LAMP Stack”

In dieser Übung wollen wir unser Blog CMS um eine asynchron geladene Listendarstellung unserer Blogbeiträge erweitert. Diese Übung baut auf Übung 8 auf, da zu ihrer Durchführung ein Webservice für den Abruf von Blogbeiträgen notwendig ist.

1. Template Umbau

Das Template das bisher die Darstellung der Beiträge übernimmt, muss angepasst werden. Bisher war ein einfaches, direkt in HTML umgesetztes Template vorhanden. Das muss jetzt in ein Template umgewandelt werden, das nach Bedarf auf Clientseite in den DOM integriert werden kann. Hierfür ist das HTML *template* Element sinnvoll: https://www.w3schools.com/TagS/tag_template.asp

1. Zerlege die bisherigen Templates und Views der aktuellen Darstellung und breche sie auf eine View und ein HTML Template herunter.
    a. Ergebnis sollte eine View Klasse sein, die ein Template einbindet das sich rein um die Darstellung kümmert.
2. Integriere eine prototypische HTML Darstellung eines einzelnen Blogbeitrags mithilfe des HTML *template* Elements.
    a. Beim Überprüfen der Funktionalität solltest du eine leere Darstellung erhalten, da das *template* Element erst vom Javascript genutzt werden muss.

2. Asynchrone Anfrage

Nun gilt es die anzuzeigenden Inhalte per Webservice zu laden und in der richtigen Stelle im DOM einzubinden. Nutze dazu die Javascript Konzepte von Promises und der *fetch()* Funktion: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch

1. Erstelle eine Javascript Datei und binde sie so ein, dass sie sinnvoll in deinem Template Platz findet.
    a. Überlege dir auf jeden Fall wie du ein Blockieren des Renderings vermeiden kannst.
2. Erstelle einen Promise, in dem du die Blogbeiträge als JSON lädst.

   a. Nutze den *fullfil* Callback des Promise um die geladenen Daten in den DOM zu integrieren. Das W3Schools Beispiel zur Nutzung des *template* Tags kann dir zeigen wie: https://www.w3schools.com/TagS/tryit.asp?filename=tryhtml5_template3
3. Integriere eine Fehlerbehandlung, die bei Nichterhalt der angefragten Daten oder bei Fehler an den Nutzer zurückmeldet.
   a. Nutze hierzu den *reject* Callback des Promises.
   b. Um einen Fehler als solchen auch erkennen zu können, muss dein Webservice mit den entsprechenden HTTP Status Codes arbeiten!

Bonus: Baue einen Spinner ein, der das Laden der Beiträge für den Nutzer ersichtlich macht.