

# Projekt: Shop

jan.schulz@devugees.org

# Vorbereitungen

- 1 – Den letzten Stand von [github.com/foobaroo/fbw14](https://github.com/foobaroo/fbw14) pullen oder klonen.
- 2 – Kopiere den Ordner `/javascript/projekte` in Deinen lokalen Desktop Ordner.
- 3 – Gehe zu Deinem Desktop/shop und führe folgenden Befehl aus **npm install**
- 4 – Nachdem die Installation abgeschlossen ist, führe folgenden Befehl aus: **node server.js**
- 5 – Gehe zu Deinem Browser und öffne `http://localhost:3000` , dort solltest Du eine leere Website sehen.
- 6 – Gehe nun bitte in das Verzeichnis `public/` und wirf einen Blick hinein. Dort siehst Du die `index.html` and `styles.css` – Das sind die Dateien mit denen Du arbeiten wirst.

# Aufgabe 1

Implementiere das folgende Design. Beachte dabei, dass es responsive sein soll. Falls Du magst, kannst Du Bootstrap verwenden.

## **Beachte Folgendes:**

- Jedes Thumbnail-Bild ist 350x200 Pixel
- Der Nutzer soll nach Beendigung von Aufgabe 1 frei herumnavigieren/-klicken können, ohne dass die Seite neugeladen werden muss.
  - Es sollen noch keine JSON-Daten vom Server geladen werden

# MediaShop

All | Books | Movies | Music

Cart

<div>Product IMAGE</div> <div>100 €</div>	<div>Product IMAGE</div> <div>100 €</div>	<div>Product IMAGE</div> <div>100 €</div>	<div>Product IMAGE</div> <div>100 €</div>
<div>Product IMAGE</div> <div>100 €</div>	<div>Product IMAGE</div> <div>100 €</div>	<div>Product IMAGE</div> <div>100 €</div>	<div>Product IMAGE</div> <div>100 €</div>

Starting Page: Product Details -> Nachdem der Nutzer auf ein Thumbnail geklickt hat

# MediaShop

All | Books | Movies | Music

Cart

Product  
IMAGE

## Product Name

Lorem ipsum dolor sit amet, consetetur  
sadipscing elitr, sed diam nonumy eirmod  
tempor invidunt ut labore et dolore magna  
aliquyam erat, sed diam voluptua.

Quantity:  100€

Add to Cart

Starting Page: Product Details -> Nachdem der Nutzer auf “Add To Cart” (Productdetails)  
“Cart” geklickt hat.

# MediaShop

All | Books | Movies | Music

Cart

Product  
IMAGE

2 x **Product Name:** **150€**

Remove

Product  
IMAGE

2 x **Product Name:** **300€**

Remove

Product  
IMAGE

2 x **Product Name:** **50€**

Remove

---

**Total:** **500€**

Buy Now

Starting Page: Product Details -> Nachdem der Nutzer auf “Buy Now“ geklickt hat

# MediaShop

[All](#) | [Books](#) | [Movies](#) | [Music](#)

[Cart](#)

Thanks for your purchase!

# MediaShop

Books | Movies | Music

Product  
IMAGE

100 €

Product  
IMAGE

100 €

Product  
IMAGE

100 €

Product  
IMAGE

100 €



# MediaShop

Books | Movies | Music



100€

Add to Cart

Lorem ipsum dolor sit  
amet, consetetur  
sadipscing elitr, sed  
diam nonumy eirmod  
tempor invidunt ut  
labore et dolore vero

# MediaShop

Books | Movies | Music



2 x **Product Name:** 150€

Remove



Total: **150€**

Buy Now

# MediaShop

Books | Movies | Music

Thanks for your  
purchase!

# Aufgabe 2

1. Starte Postman und erstelle 3 Requests “loadProducts“, “loadProductsByCategorie“ und “postOrder“.
2. Modifiziere die Requests aus 1.) entsprechend den folgenden URLs und ggfalls. Parametern:

loadProducts: GET <http://localhost:3000/products>

loadProductsByDetail: GET <http://localhost:3000/products>?category=Movies

postOrder: POST <http://localhost:3000/order>

3. Analysiere die Antworten, welche Du bei Postman zurückbekommst. Entnehme v.a. daraus, welchen Body Du bei postOrder senden musst, um einen Order im System anzulegen.
4. Im Shop Verzeichnis, erstelle ein Verzeichnis /experiments und erstelle darin eine leere Website mit index.html und main.js
  - Erstelle drei Buttons mit den Beschriftungen “Load Products“, “Load Products by Detail“ und “Post Order“.
  - Erstelle für jeden der drei Buttons jeweils einen Fetch-Request (AsyncAwait) entsprechend dem jeweiligen Postman-Request und gib alle Outputs auf der Console aus.

# Aufgabe 3

Mit Hilfe der Aufgabe 2 gebauten Experimente, erweitere Deinen Shop mit Fetch und AsyncAwait.

1. Lade alle Produkt-Informationen mit Hilfe eines Fetch-Aufrufs `loadProducts`, stelle die Thumbnails entsprechend dar sowie die Produkt-Details. Jedes mal, wenn der Nutzer die Seite neu lädt oder auf das Logo oder All klickt, sollen alle Produkte per Fetch neugeladen und angezeigt werden.
2. Lade alle Product-Informationen abhängig von der Kategorie mit Hilfe eines Fetch-Aufrufs ähnlich wie `loadProductsByDetail`. Lade die jeweilige Kategorie wenn der Nutzer auf Books, Movies oder Music klickt.

# Aufgabe 4

Der Shop soll nun die Warenkorb-Funktionalität bekommen.

1. Erstelle ein neues globales Array "cart".
2. Wenn der Nutzer auf "Add to Cart" klickt, soll ein neues Objekt in das cart-Array gepusht werden. Beachte hierbei, dass jedes Item eine Id hat.
3. Wenn der Nutzer zum Warenkorb geht, sollen alle Items angezeigt werden, die der Nutzer schon in den Warenkorb hineingetan hat. Mehrfache Items sollen durch die Quantity beschrieben werden. Z.B. Wenn der Nutzer 2x Beethoven in den Warenkorb getan hat, soll die Quantity bei 2 sein.
4. Wenn der Nutzer im Warenkorb ist und auf den Remove-Button klickt, soll das Item (auch mehrfache Items) aus dem Warenkorb getan werden. Beachte, dass hier auch die localStorage verändert werden muss.

# Hilfreiches

- Kann ganz hilfreich sein
  - `Delegateexample.html` – Klick Events für Elemente, die dem DOM-Tree dynamisch hinzugefügt wurden