FER3

Preddiplomski studij

Računarstvo

Razvoj programske potpore za web

Druga laboratorijska vježba – JavaScript

Ak. god. 2020./2021.

1 Uvod

Druga laboratorijska vježba nadovezuje se na prvu laboratorijsku vježbu. U sklopu ove vježbe bit će potrebno na osnovu priloženih JavaScript primjera napraviti pripremu koja je potrebna za rad tijekom laboratorijske vježbe. Na samoj vježbi potrebno je izmijeniti pripremu, odnosno potrebno je dodati nove ili modificirati postojeće funkcionalnosti.
Napomena: Kanal za sva pitanja vezana uz 2. laboratorijsku vježbu na Teamsu je ovdje.

1.1 Razvojno okruženje

Preporučuje se korištenje editora Visual Studio Code.
Korisna ekstenzije:

- Alat za automatsko formatiranje koda Beautify
- Alat za jednostavnije pisanje HTML koda Emmet - već dostupan u editoru.

Za debugging koristite:

- Chrome DevTools
- Firefox Developer Tools

1.2 Organizacija projekta

Slika 1 prikazuje strukturu projekta. Sve datoteke (odnosno projekt) priložene su na poveznici.
Napomena: Nije potrebno dodavati nove datoteke.

```
root
├── fonts --> Fontovi
│   ├── DancingScript-Bold.woff
│   ├── DancingScript-Bold.woff2
.
.
.
│   ├── Roboto-Thin.woff
│   ├── Roboto-Thin.woff2
├── images  --> Slike
│   ├── calendar-icon.png
│   ├── chrysanthemum.jpg
.
.
.
│   ├── tulip.jpg
│   ├── wild-flowers.jpg
├── index.html --> potrebno dodati JS
├── order.html --> potrebno izmijeniti JS
└── styles --> Vanjske knjižnice
    └── main.css
    └── order.css
└── vendor --> Vanjske knjižnice
    └── normalize.css
```

Slika 1: Struktura projekta za 2. laboratorijsku vježbu

2 Priprema

Vaš zadatak zadatak za pripremu je izmijeniti datoteku *order.html* odnosno dodati JavaScript kôd koji nedostaje označen s `INSERT CODE HERE` tako da se prikazuju proizvodi po kategorijama. Uz implementaciju navedene funkcionalnosti potrebno je riješiti zadatak iz podpoglavlja 2.1, koji je primjer kompleksnosti zadataka koji će biti traženi za vrijeme vježbi. Samo uz rješavanje kompletne pripreme moguće je pristupiti riješavanju zadataka za vrijeme vježbi.
U datoteci *order.html* implementiran je dohvat podataka o kategorijama (Primjer 1) pomoću zadanog URL-a u obliku JSON-a (Primjer 2) i prikaz naziva kategorija na stranici. U primjeru 1 dohvat i prikaz su ostvareni manipulacijom DOM-a pomoću JavaScript kôda koristeći predložak (*template*) s identifikatorom "category-template" prikazan na primjeru 3.

```
<script>
    let getData = async function() {
        let response = await fetch("https://web1lab2.azurewebsites.net/categories");
        let data = await response.json();
        addCategories(data);
    }

    let addCategories = async function (categories) {
        let main = document.querySelector('main');
        let categoryTemplate = document.querySelector('#category-template');

        for (let index = 0; index < categories.length; index++) {
            let category = categoryTemplate.content.cloneNode(true);
            let categoryTitleElement = category.querySelector('.decorated-title > span');
            categoryTitleElement.textContent = categories[index].name;
            // INSERT CODE HERE --> PRIPREMA

            // END INSERT --> PRIPREMA
            main.appendChild(category);
            }
    };
    getData();
</script>
```

Primjer 1: JavaScript kôd za dohvat i prikaz kategorija

```
[
  {
    "id": 1,
    "name": "Flowers"
  },
  {
    "id": 2,
    "name": "Tools"
  },
  ...
]
```

Primjer 2: JSON podaci o kategorijama

```
<template id="category-template">
    <section class="section secondary-color-bg">
        <div class="centered-container">
            <h1 class="decorated-title font-secondary main-color">
            <span class="main-color-emphasized">Category name</span>
                    Inventory</h1>
            <div class="separator"></div>
        </div>
        <div class="gallery">
        </div>
    </section>
</template>
<template id="product-template">
    <div class="photo-box" data-id="">
        <img class="photo-box-image" src=""></img>
        <div class="photo-box-title font-secondary main-color-emphasized">Product name</div>
        <div class="cart-btn" onclick=""></div>
    </div>
</template>
```

Primjer 3: HTML predlošci za kategorije i proizvode

Potrebno je na identičan način pomoću JavaScript kôda za svaku kategoriju dohvatiti naziv proizvoda te njegovu sliku u obliku URL-a. Dohvat proizvoda u pojedinoj kategoriji u obliku JSON-a ostvaruje se pomoću URL-a:

`https://web1lab2.azurewebsites.net/products?categoryId=1`

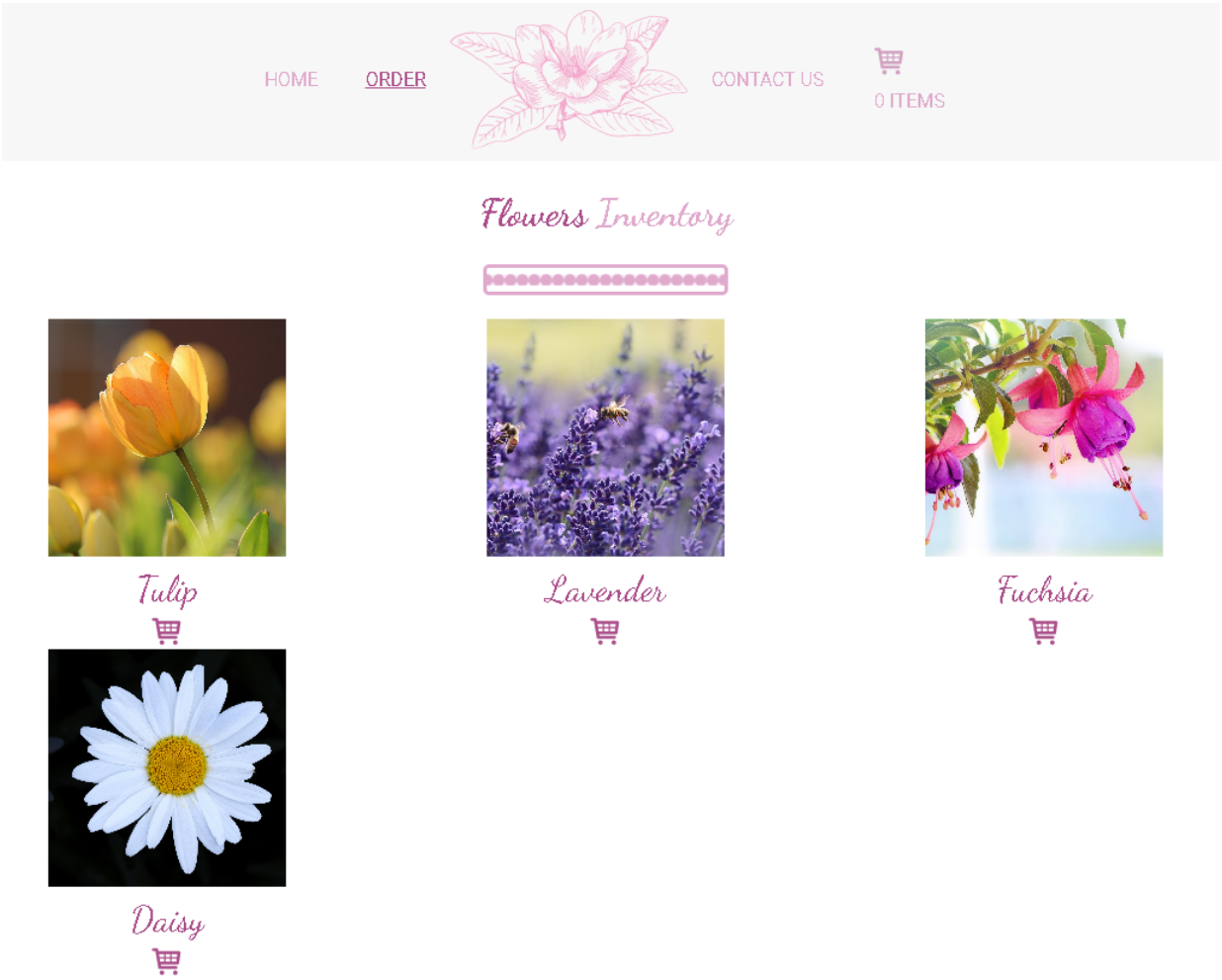
U prikazanom URL-u broj 1 je dan kao primjer za identifikator kategorije, u ovom slučaju *Flowers*. JSON podaci za kategoriju su prikazani na primjeru 4.

```
[
  {
    "id": 1,
    "name": "Tulip",
    "price": 10,
    "categoryId": 1,
    "imageUrl": "../images/tulip.jpg"
  },
 ...
]
```

Primjer 4: JSON podaci o proizvodima za kategoriju *Flowers*

Nakon dohvata podataka, manipulacijom DOM-a pomoću predloška s identifikatorom "product-template" potrebno je za svaku kategoriju prikazati njene proizvode. Konačan rezultat pripreme prikazan je slikom 2.

Napomena: Oba predloška iz primjera 3 su sadržana unutar datoteke *order.html*. Nije potrebno pisati dodatan HTML kôd niti definirati nove stilove. Potrebni stilovi su definirani i referencirani pomoću predložaka.



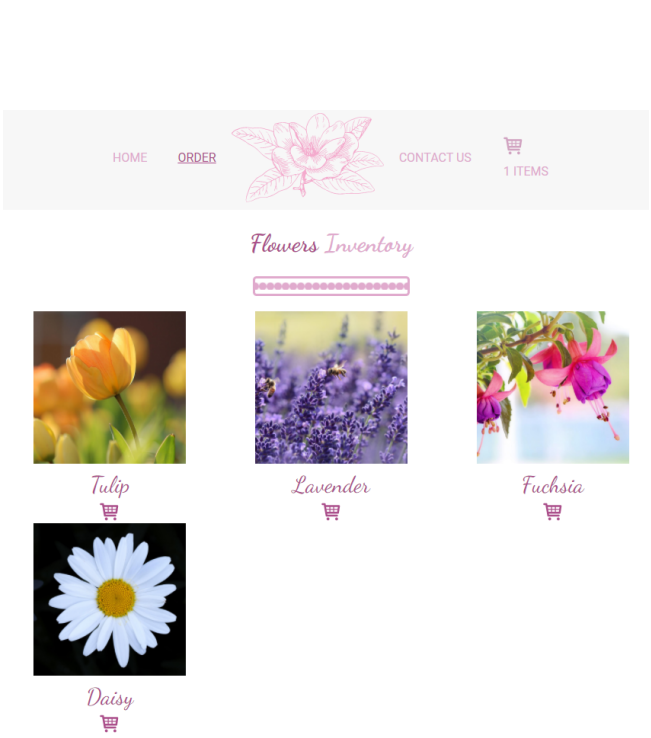
Slika 2: Stranica *order.html*

2.1 Zadatak

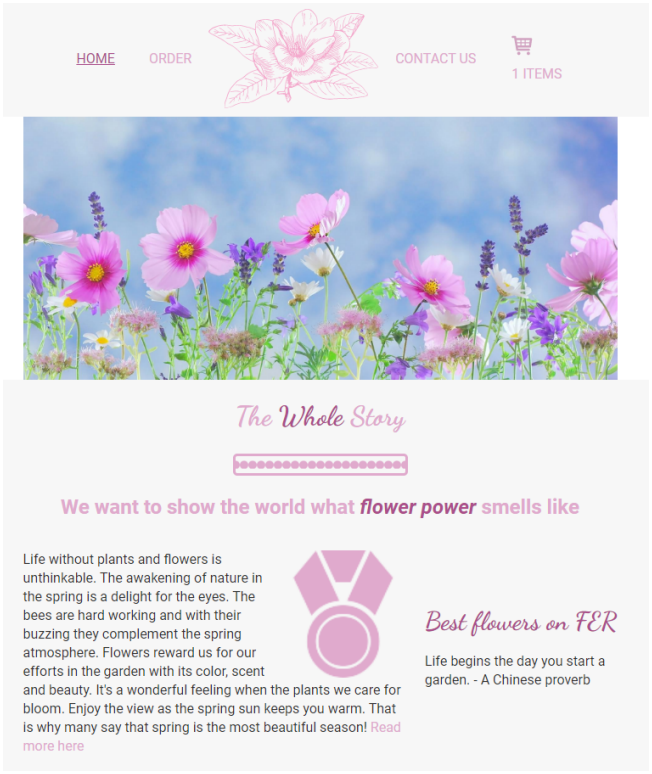
Ovu funkcionalnost POTREBNO JE implementirati u sklopu pripreme! Služi kao primjer kompleksnosti nove funkcionalnosti ili promjene postojećih koje mogu biti tražene, a trebate ih moći implementirati za vrijeme trajanja vježbe. Bit će opisana tražena funkcionalnost i slika stranice onako kako bi trebala izgledati po završetku vježbe.

Zadatak je implementirati promjenu brojčanika košarice tako da se prilikom klika na simbol košarice ispod proizvoda implementira dodavanje proizvoda u košaricu. Košaricu je potrebno ostvariti pomoću lokalnog spremnika (engl. *local Storage*) tako da se za svaku stavku košarice pohranjuje identifikator proizvoda i trenutna količina. Brojčanik uz košaricu prikazuje ukupan broj proizvoda u košarici, npr. ako su u košarici dodana dva tulipana i jedna lopata, brojčanik će prikazivati broj 3. Vrijednost brojača je potrebno očuvati prilikom promjena stranica, npr. odlaskom na *index.html* potrebno je prikazati isti broj koji je bio na stranici *order.html*.

Napomena: Za vrijeme laboratorijske vježbe bit će potrebno ostvariti sličnu funkcionalnost.



(a) klik na košaricu proizvoda uvećava brojčanik košarice (npr. 1 ITEMS)



(b) stranica **index.html** zadržava istu vrijednost brojčanika

Slika 3: Primjer zadane funkcionalnosti na laboratoriju

3 Često postavljana pitanja

Je li nužno pisati JS kôd unutar granica zadanih komentarima?
Zadanu funkcionalnost nije nužno pisati unutar zadanih granica označenih komentarima. Zbog jednostavnosti predviđeno je pisanje u HTML datotekama, ali moguće je napisati rješenje u odvojenoj JS datoteci.

Treba li se prilikom ponovnog otvaranja internetskog preglednika i web stranice prikazati stari broj proizvoda u košarici ili taj broj trebamo resetirati na 0?
Nije potrebno resetirati na 0, ostaviti stari broj proizvoda.

Može li se zadatak s košaricom riješiti bez korištenja localStorage-a
Teoretski, ako baš ne želite localStorage, možete koristiti sessionStorage ili IndexedDB. Savjet je općenito držite se KISS principa.

Treba li se u order.html dodavati kategorija ako u JSON datoteci nema ni jedan proizvod za nju?
Nije potrebno provjeravati postoje li proizvodi u kategoriji.

Različite kategorije ne moraju imate pozadine različite boje kao u prvoj laboratorijskoj vježbi?
Ne moraju.

U internetskom pregledniku "onclick" ne funkcionira. Imate li ideju kako ovo riješiti?
Provjerite konzolu u Developer toolsima i provjerite greške. Također, proučite reference i primjere (npr., za setAttribute) na poveznicama navedenima na kraju dokumenta.

4 Asinkrona funkcija (async/await)

U 3. predavanju iz JS obrađeno je gradivo vezano za asinkrone funkcije i *Promise* objekte. Za potrebe 2. laboratorijske vježbe kod dohvata podataka s poslužitelja potrebno je koristiti ključne riječi *async* i *await*. Funkcije označene s *async* su asinkrone što znači da ne blokiraju izvršavanje JS kôda, dok ključna riječ *await* omogućuje sinkrono ponašanje funkcija. Važno je kako se *await* koristi samo unutar asinkronih funkcija.

Zašto ne koristimo sinkrone ("obične") funkcije?
Kod dohvata podataka iz pripreme unutar funkcije `getData` (Primjer 5) koristi se funkcija `fetch` koja je asinkrona. Kako želimo u `response` varijablu pohraniti rezultat funkcije `fetch` dodana je ključna riječ *await*. Kada to ne bismo napravili, izvršavanje skripte bi se nastavilo, samo bi bilo pozvano dohvaćanje podataka, dok bi varijabla `response` sadržavala objekt koji ne sadrži rezultat dohvata. Po asinkronom ponašanju rezultat bi bio dohvaćen i spremljen u objekt `response` kasnije dok bi internetski preglednik nastavio izvršavati JS kôd izvan funkcije `getData`. Time bi se izgubili podaci o kategorijama koji ne bi mogli biti prikazani pomoću DOM-a u funkciji `addCategories` i dogodila bi se greška. Slično je s asinkronom funkcijom `response.json()` za dohvat JSON rezultata. Kako se *await* može koristiti samo u asinkronim funkcijama, funkcija `getData` je definirana kao asinkrona.

```
<script>
    let getData = async function() {
        let response = await fetch("https://web1lab2.azurewebsites.net/categories");
        let data = await response.json();
        addCategories(data);
    }
</script>
```

Primjer 5: Asinkrona funkcija `getData`

5 Kako pisati JavaScript?

Obavezno je proučiti predavanja vezana za JavaScript. Kao pripremu prepručeno je pogledati i samostalno napraviti korak po korak igru CookieClicker. Implementacija igre obuhvaća modeliranje i programiranje u JS, manipulaciju DOM-a, lokalni spremnik i JSON, gradivo koje je potrebno savladati za uspješno rješavanje zadataka vježbe. *CookieClicker* implementiran je pomoću objektno-orijentirane paradigme, ali u laboratorijskoj vježbi to nije potrebno.

Također preporučeno je proučiti upute i primjere na sljedećim poveznicama:

- Atributi
- JSON
- HTML predlošci (*template*)
- Lokalni spremnik
- Imenovanje klasa DOM elemenata za JS