



FER3

Preddiplomski studij

Računarstvo

Razvoj programske podpore za web i pokretne uređaje

Druga laboratorijska vježba – JavaScript
– modifikacija pripreme

1 Uvod

Uz izradu pripreme, potrebno je modificirati pripremu. Modifikacija pripreme sadrži zajednički dio (košarica) i varijaciju modifikacije. Odredite koju varijaciju nad pripremom imate prema datoteci na poveznici [zadaci](#). U datoteci su sadržani JMBAG i redni broj varijacije, npr. studentu s JMBAG-om 0036511119 određen je 1. zadatak:

- 0036511119 1

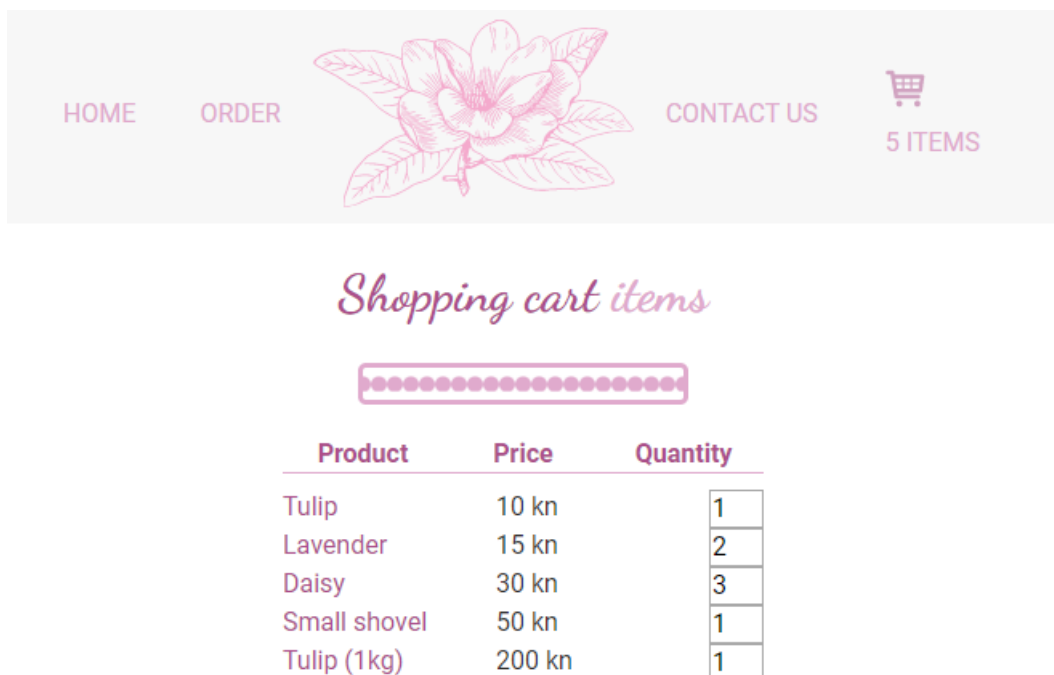
2 Zadatak košarica

2.1 Brojčanik košarice i dodavanje proizvoda

Zadatak promjene brojčanika košarice potrebno je modificirati tako da se prilikom klika na košaricu proizvoda implementira dodavanje proizvoda u košaricu. Košaricu je potrebno pohraniti u lokalnoj pohrani tako da se za svaku stavku košarice pohranjuje identifikator proizvoda i trenutna količina. U odnosu na pripremu potrebno je promijeniti prikaz tako što će brojčanik uz košaricu prikazivati broj različitih proizvoda u košarici, npr. ako su u košarici dodana dva tulipana i jedna lopata, brojčanik će prikazivati broj 2. Vizualan primjer za 5 proizvoda je vidljiv na slici 1.

2.2 Implementacija stranice s pregledom košarice

Potrebno je implementirati stranicu **cart.html** za pregled košarice stiliziranu prema slici 1. Stranica **cart.html** otvora se prilikom klika na košaricu u izborniku koji vodi na stranicu košarice iz bilo koje stranice dostupne u izborniku. Pretpostavka za prikaz stranice je lokalno pohranjena košarica iz zadatka zadanog u poglavlju 2.1.



Slika 1: Stranica za pregled košarice **cart.html**

Za prikaz stavki košarice potrebno je napisati JavaScript kôd uključen u stranicu **cart.html** za dohvat podataka o proizvodu. Dohvat proizvoda prema identifikatoru u obliku JSON-a ostvaruje se pomoću URL-a:

<https://wimlab2.azurewebsites.net/products/1>

U prikazanom URL-u broj 1 je dan kao primjer za identifikator proizvoda, u ovom slučaju *Tulip*. JSON podaci za proizvod su prikazani na primjeru 1.

```
{
  "id": 1,
  "name": "Tulip",
  "price": 10,
  "categoryId": 1,
  "imageUrl": "https://i.imgur.com/Ttir6mp.jpg"
}
```

Primjer 1: JSON podaci o proizvodu *Tulip*

Nakon dohvata podataka, manipulacijom DOM-a pomoću gotovih predložaka prikazanih na primjeru 2 potrebno je prikazati zaglavlje i stavke košarice. Stavka košarice se sastoji od naziva proizvoda, njegove cijene i trenutne količine u košarici.

```
<template id="cart-template-header">
  <div class="cart-header">
    <div class="cart-header-title">Product</div>
    <div class="cart-header-title">Price</div>
    <div class="cart-header-title">Quantity</div>
  </div>
</template>
<template id="cart-template-item">
  <div class="cart-item">
    <div class="cart-item-title">Tulips</div>
    <div class="cart-item-price">15 kn</div>
    <input type="number" class="cart-item-quantity"></input>
  </div>
</template>
```

Primjer 2: HTML predlošci za zaglavlje i sadržaj košarice

3 Varijacija modifikacije

3.1 Zadatak 1. - loader

U slučaju kada dohvat podataka dugo traje, stranica može izgledati kao da je blokirana. Time degradiramo korisničko iskustvo. Vaš je zadatak implementirati loader koji korisniku daje do znanja da se podaci učitavaju, a nestaje kada se podaci učitaju.

Budući da naš poslužitelj nije spor, morat ćemo simulirati kašnjenje. To možemo postići tako da kašnjenje u milisekundama predamo kao delay parametar u GET zahtjev. Primjeri za kašnjenje od 3 sekunde:

```
{
  https://wimlab2.azurewebsites.net/products?delay=3000&categoryId=1
  https://wimlab2.azurewebsites.net/categories?delay=3000
}
```

Napomena: Dodajte kašnjenje od MINIMALNO 3 SEKUNDE u SVAKI zahtjev na naš poslužitelj, u suprotnom je moguće da će poslužitelj toliko brzo odgovarati da će loader biti jako kratko vidljiv!

Na raspolaganju vam je dan gotov loader i slobodno koristite ovaj HTML kod:

```
<div class="loader">
  <div class="loader-text main-color-emphasized font-secondary">Molimo pricekajte</div>
  <div class="loader-pulsar"></div>
</div>
```

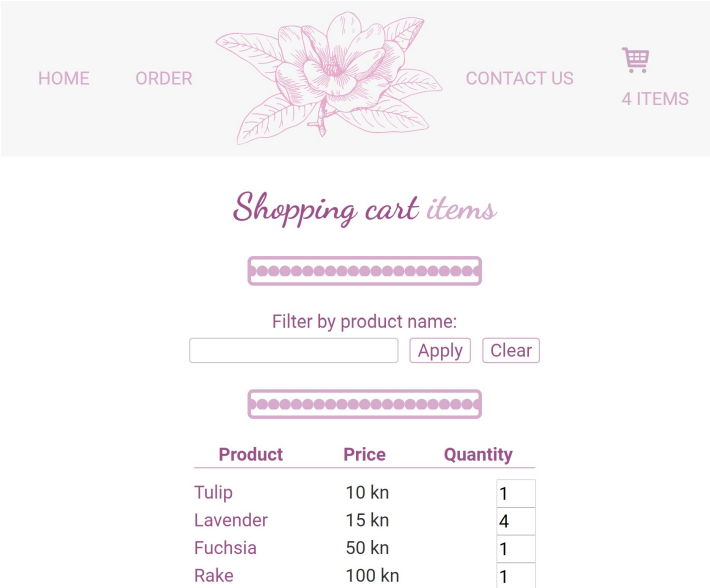
Osim toga dostupna je klasa `hidden` koja skriva loader. Potrebno je centrirati loader, inicijalno ga prikazati odnosno prikazivati dok se dohvaćaju podaci, te kada se **svi** podaci dohvate sakriti dodavanjem klase `hidden` na loader. Za vizualizaciju, pogledajte priloženi video.

3.2 Zadatak 2 - filter

Zadatak je dodati jednostavni filter s kojim se elementi košarice filtriraju po nazivu. Npr.:

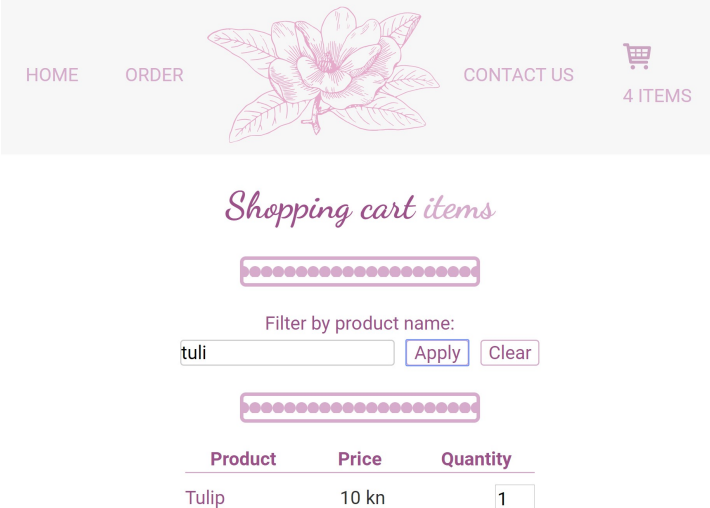
- ako u košarici postoje artikli "Tulip", "Lavender" i "Rake", a u filter se upiše string "tul" i klikne "Apply", prikazat će se samo artikal "Tulip" pretraživanje je "case insensitive" i traži se substring!).
- ako se nakon toga u filter upiše "A" i klikne "Apply", "Tulip" će nestati s popisa, a prikazat će se "Lavender" i "Rake".
- klikom na gumb "Clear" pojavit će se svi artikli, a polje za unos filtera će se obrisati.

Korištenje filtera ne smije uzrokovati ponovno učitavanje stranice ili dohvat podataka sa servera. Izgled i raspored komponenti moraju biti kao na slici.



Product	Price	Quantity
Tulip	10 kn	1
Lavender	15 kn	4
Fuchsia	50 kn	1
Rake	100 kn	1

(a) prikaz filtera



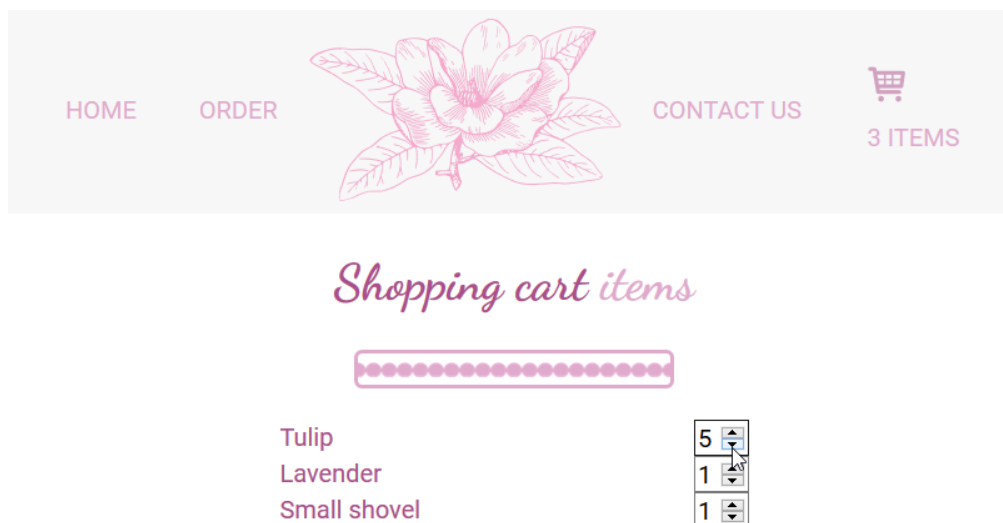
Product	Price	Quantity
Tulip	10 kn	1

(b) primjena filtera za izraz "tuli"

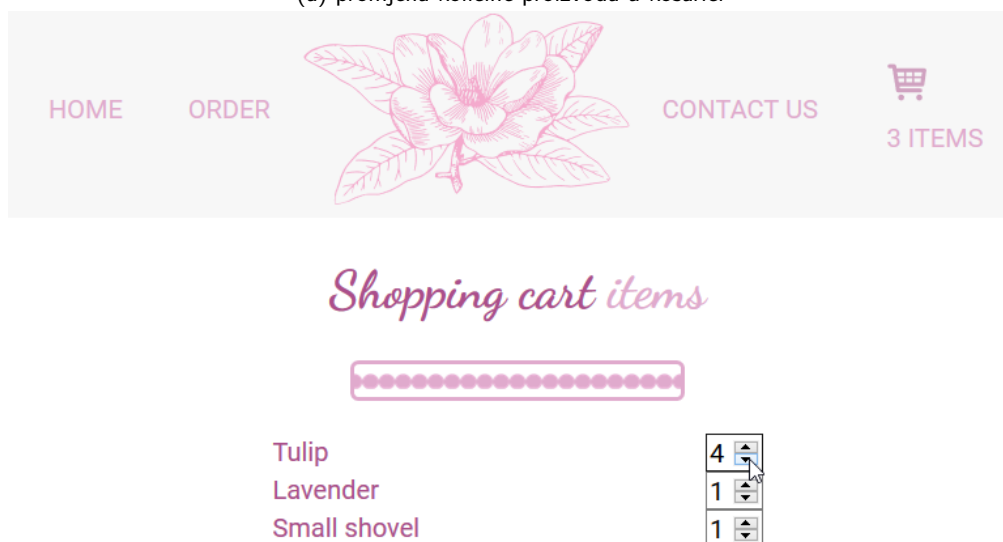
Slika 2: Zadatak 3. - ishod

3.3 Zadatak 3. - izmjena količine proizvoda na košarici

Vaš je zadatak implementirati mogućnost izmjene količine pojedinog artikla u košarici, na stranici **cart.html**. Izmjena treba biti implementirana na ergonomski način, bez dodatnog gumba za spremanje promjena.



(a) promjena količine proizvoda u košarici



(b) prikaz **cart.html** nakon promjene količine i osvježavanja stranice

Slika 3: Zadatak 3. - ishod

3.4 Zadatak 4. - brisanje proizvoda i košarice

Vaš zadatak je implementirati brisanje proizvoda iz košarice. Morate omogućiti brisanje pojedinog proizvoda ili brisanje sadržaja cijele košarice odjednom (pogledati priloženi video!):

- pritiskom na gumb "x" pored pojedinog proizvoda, briše se samo taj proizvod iz košarice
- pritiskom na gumb "Empty cart" na dnu košarice briše se sadržaj cijele košarice (uključujući prvi red tablice s obzirom na to da je košarica prazna)

Nakon brisanja NE učitavati ponovno cijelu stranicu da bi vidjeli novo stanje.

Za gumb brisanja pojedinog proizvoda i gumb brisanja sadržaja cijele košarice koristiti gotove CSS klase i HTML predloške:

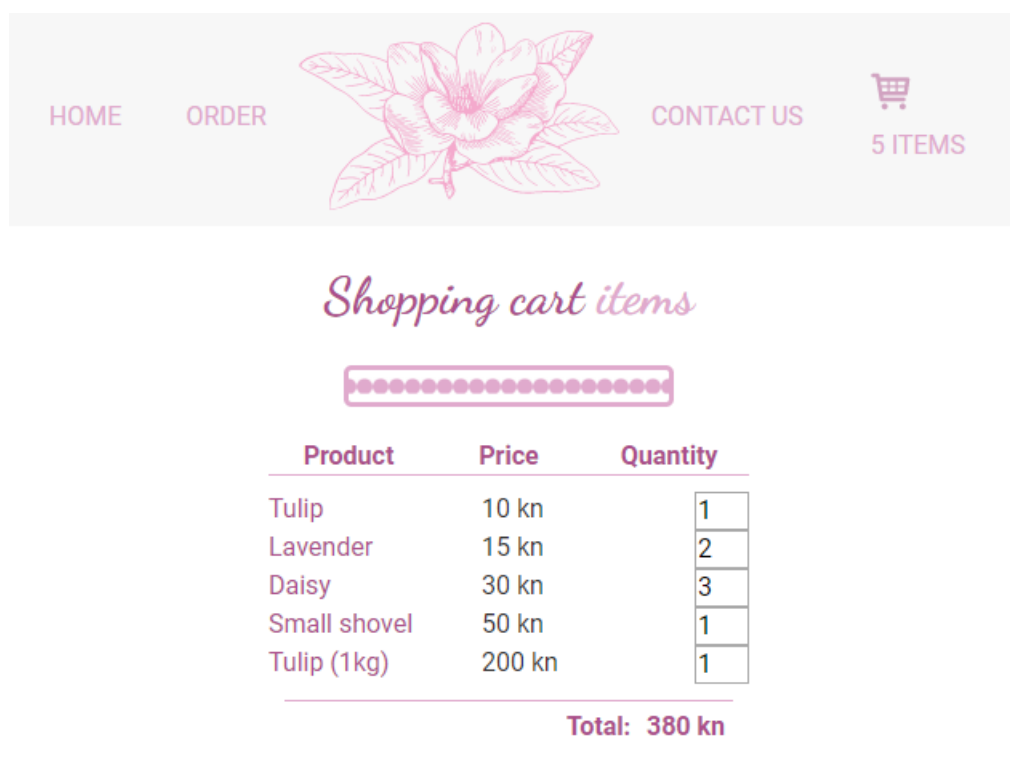
```
<template id="cart-template-item">
  <div class="cart-item">
    <div class="cart-item-title">
      Tulips
    </div>
    <div class="cart-item-price">
      15 kn
    </div>
    <input type="number" class="cart-item-quantity">
    <input type="button" class="cart-btn-x-remove-item-btn" value="x">
  </div>
</template>
<template id="cart-template-footer">
  <input type="button" style="float: right;" class="cart-btn-x-remove-all-btn" onclick="emptyCart()"
  value="Empty cart">
</template>
```

Primjer 3: HTML predložak za brisanje košarice

Za prikaz funkcionalnosti vidjeti priloženi video.

3.5 Zadatak 5. - Ukupan iznos košarice

Zadatak je izračunati i prikazati ukupan iznos košarice. Pri tome imati na umu kako količina određenog proizvoda utječe na konačan iznos. Funkcionalnost ukupnog iznosa košarice je potrebno implementirati prema zadanoj slici (Slika 4). Za prikaz možete koristiti gotove CSS klase i HTML predložak koji je prikazan na primjeru 4.



Slika 4: Zadatak 5. - ishod

```
<template id="cart-total-template">
  <div class="cart-total">
    <div>Total:</div>
    <div id="cart-total-value"></div>
  </div>
</template>
```

Primjer 4: HTML predložak za ukupan iznos košarice