UNIVERZITET U BEOGRADU
MATEMATIČKI FAKULTET

Aleksandar Milosavljević

RAZVOJ "PURE" OKRUŽENJA ZA RAZVOJ VEB INTERFEJSA

master rad

Beograd, 2021.

Mentor:

dr Saša Malkov, vanredni profesor
Univerzitet u Beogradu, Matematički fakultet

Članovi komisije:

dr Aleksandar Kartelj, docent
Univerzitet u Beogradu, Matematički fakultet

dr Ivan Čukić, docent
Univerzitet u Beogradu, Matematički fakultet

Datum odbrane: 15. januar 2016.

Svima osim LaTex-u

Naslov master rada: Razvoj "Pure" okruženja za razvoj veb interfejsa

Rezime: Rezime placeholder

Ključne reči: veb, internet, interfejs, html, css, javaskript, razvojno okruženje

Sadržaj

Glava 1

Uvod

Poboljšanje infrstrukture Internet mreže, to jest, povećanjem brzine i stabilnosti internet komunikacije, naša interakcija sa računarima danas izgleda značajno drugačije nego pre 20 godina. Umesto stonih računara sa velikim kućištima i glasnim rashladnim sistemima, današnji najrasprostranjeniji računari nam staju u džep. Koristimo ih kako bi komunicirali sa bližnjima i bili u toku sa svetskim dešavanjima.

Iako današnji mini računar koji gotovo svi nosimo u džepu ima više procesorske moći nego računar koji je korišćen tokom Appolo misije(ref), danas te lične računare ne koristimo za teška izračunavanja i komplikovane procedure. Koristimo ih najčešće kao terminale kako bismo pristupili internet resursima, to jest, podacima koji se nalaze na velikim centralizovanim računarskim sistemima. Veb interfejs je tako postao glavni način naše interakcije sa računarom i internetom.

Posledica toga je značajna promena fokusa u svetu razvoja aplikativnog softvera. Ukoliko pogledamo rezultat upitnika kompanije „Stack Overflow" (ref)(`https://insights.stackoverflow.com/survey/2020#technology-programming-scripting-and-mark`
o najkorišćenijim programerskim alatima, videćemo da prva dva mesta na listi zauzimaju upravo veb tehnologije (Na prvom mestu je JavaScript sa 69.7%, dok je na drugom HTML/CSS sa 62.4%).

Kako potreba za novim softverom daleko prevazilazi mogućnost softverskih inženjera da taj softver isporuče, biblioteke i okruženja koja pomažu pri razvoju softvera postali su veoma važan deo inženjerskog alata. U domenu razvoja veb aplikacija, na tržištu se izdvajaju tri razvojna okruženja. Ova tri razvojna okruženja najviše se međusobno razlikuju po pristupu u promeni stanja aplikacije i razumevanju toga šta je centralni deo dizajna (u funkcionalnom smislu) jedne veb

aplikacije.

Kada posmatramo ova popularna razvojna okruženja, uočavamo dva bitno različita pristupa u arhitekturi:

1. *Model-Pogled-Kontroler* šablon u kom aplikacija predstavlja skup povezanih komponenti organizovanih u hijerarhisku strukturu, kojima je pridruženo stanje i ponašanje

2. Flux šablon u kom se aplikacija posmatra kao mašina stanja čija je vizuelna reprezentacija samo rezultat trenutnog stanja.

Prvi pristup je prisutan u okruženju Angular, drugi je dominantan u biblioteci React, dok Vue.js zastupa dizajn koji predstavlja kompromis između ova dva pristupa.

Kroz ovaj rad ispratićemo razvoj jednog modernog razvojnog okruženja za izradu SPA veb aplikacija.

Glava 2

Arhitekture SPA aplikacija

2.1 Model-Pogled-Kontroler šablon

2.2 Flux šablon

Razrada placeholder

Glava 3

Problemi sa popularnim okruženjima

3.1 Upravljanje stanjem

3.2 Razdvajanje odgovornosti

Glava 4

Okruženje Pure

Kako bismo se upoznali detaljnije sa novim okruženjem, krenućemo od jednostavnog primera, *Hello World (eng. Zdravo Svete)* aplikacije.

4.1 Podešavanje radnog okruženja

Okruženje *Pure* može se preuzeti kroz *npm*[1] sistem. To možemo uraditi na dva načina:

1. Instaliranjem paketa `pure-framework` u već postojeći *npm* modul.
2. Pokretanjem skripte za pravljenje `hello-world` projekta, bez prethodnog instaliranja *npm* paketa.

Ukoliko već imamo svoj *npm* modul u kom pišemo kod, možemo iskoristiti prvi način i instalirati `pure-framework` komdanom 4.1:

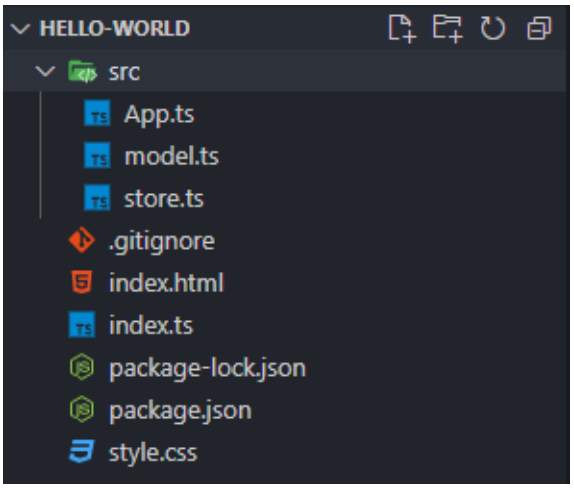
```
$ npm install pure-framework                                    (4.1)
```

Ovde ćemo, jednostavnosti radi, koristiti drugi pristup u kojem krećemo sa praznim folderom, u kom ćemo pomoću *npx* skripte napraviti početnu „Zdravo Svete" aplikaciju. Potrebno je da se u `shell`-u pozicioniramo u prazan direktorijum i da odatle pokrenemo komandu 4.2:

```
$ npx pure-framework hello-world                                (4.2)
```

[1]Upravljač paketa za Node *(eng. Node Package Manager)*

Ova komanda će napraviti novi direktorijum sa nazivom `hello-world` i u njega klonirati repozitorijum sa minimalnom *Pure* aplikacijom. Ukoliko je skripta izvršena bez grešaka, direktorijum `hello-world` bi trebalo da sadrži fajlove prikazane na slici 4.1.



Slika 4.1: Grafikon

Kako bismo pokrenuli našu aplikaciju, potrebno je da se pozicioniramo u novonapravljeni direktorijum i instaliramo neophodne *npm* pakete, izvršavanjem komande prikazane na isečku 4.3:

```
$ cd hello-world && npm install                                    (4.3)
```

Nakon što se ova komanda uspešno izvrši, možemo da pokrenemo našu aplikaciju komandom prikazanom na isečku 4.4:

```
$ npm run start                                                    (4.4)
```

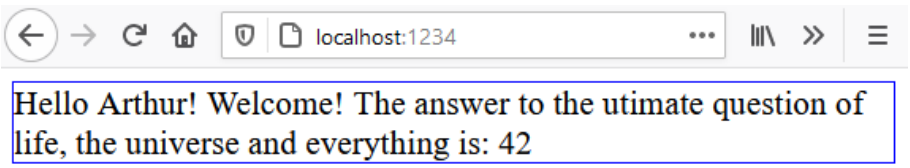
Ukoliko pokretanje ove komande nije izbacilo grešku, konzola bi trebala da nas obavesti da je server za razvoj pokrenut i da možemo da mu pristupimo na url adresi `http://localhost:1234` (Isečak 4.1)

```
> pure-framework-hello-world-app@1.0.0 start
> parcel index.html
```

```
Server running at http://localhost:1234
Built in 27ms
```

Izlaz konzole 4.1: konzolna poruka nakon pokretanja aplikacije

Ukoliko u pretraživaču otvorimo url stranicu `http://localhost:1234`, trebalo bi da vidimo stranicu koja izgleda kao na slici 4.2



Slika 4.2: Hello World aplikacija, pokrenuta na lokalnom serveru.

4.2 Struktura aplikacije

Pogledajmo sada sadržaj fajlova koje smo napravili u koraku 4.2. Primetićemo da fajlovi `index.html` i `index.ts` gotovo da ne sadrže nikakvu programsku logiku ili opis strukture DOM[2] drveta. U narednim poglavljima upoznaćemo se detaljnije sa svrhom ovih fajlova, ali ćemo te detalje za sada preskočiti.

Fajl koji sadrži centralnu logiku naše aplikacije je `src/App.ts` (Isečak 4.2)

[2]Document Object Model

```
1  import { Component, componentFactory, Store } from "pure-framework/
       core";
2  import { div, span } from "pure-framework/htmlElements";
3  import { AppModel } from "./model";
4
5  class AppComponent extends Component<AppModel> {
6    template() {
7      return div({ class: 'app-root'}, [
8        span(`Hello ${this.state.name}! Welcome! `),
9        span(`The answer to the utimate question of life, the universe
             and everything is: ${this.state.answer}`)
10     ]);
11   }
12 }
13 export const app = componentFactory(AppComponent);
```

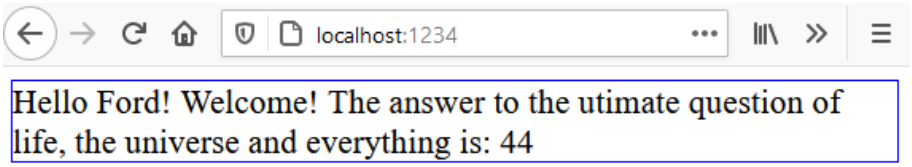
Isečak koda 4.2: Sadržaj fajla `App.ts`

Ukoliko pogledamo aplikaciju u pretraživaču (slika 4.2) videćemo da struktura elemenata DOM-a odgovara strukturi koju vraća funkcija `template()`.

Recimo da želimo da dodelimo ponašanje našoj aplikaciji, tako da svaki put kada kliknemo na element povećamo brojač za jedan i ime promenimo na „Ford". To možemo da uradimo tako što ćemo izmeniti funkciju `template()` na sledeći način:

```
6  template() {
7    return div({ class: 'app-root'}, [
8        span(`Hello ${this.state.name}! Welcome! `),
9        span(`The answer to the utimate question of life, the universe
             and everything is: ${this.state.answer}`)
10   ]).on('click', () => {
11     store.updateState({
12       answer: this.state.answer + 1,
13       name: 'Ford'
14     })
15   });
16 }
```

Isečak koda 4.3: Fajl `App.ts` nakon dodate funkcionalnosti

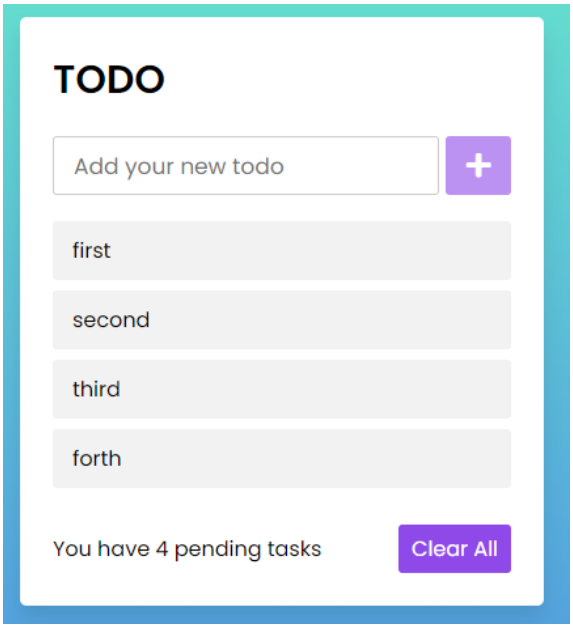


Slika 4.3: Hello World aplikacija, pokrenuta na lokalnom serveru.

Ukoliko otvorimo ponovo našu aplikaciju u pretraživaču i kliknemo na nju dva puta, trebalo bi da uočimo da broj koji se prikazuje na kraju poruke više nije 42, već 44, dok je pozdravna poruka ovog puta, umesto Arturu, namenjena Fordu (slika 4.3)

4.3 Aplikacija „Menadžer Zadataka”

Pogledaćemo sada malo kompleksniji primer. U pitanju je aplikacija za upravljanje dnevnim obavezama[3]. Online verzija ove aplikacije nalazi se na stranici `https://pure-framework-todo-demo.netlify.app/` (Slika 4.4)



Slika 4.4: TO DO aplikacija, dostupna online

[3] *(eng. TO-DO)* lista.

```
6  import { Component, componentFactory } from "./core";
7  import { button, div, header, inputText, italic, li, span, ul } from "
       ./htmlElements";
8  import { ButtonElement } from "./htmlElements/block-elements/
       buttonElement";
9
10 import { store } from "./stores/todo.store";
11 import { ToDoState } from "./models"
12
13 class ToDoListComponent extends Component<ToDoState> {
14   template() {
15     return div({ class: 'wrapper' }, [
16       ...this.headerSegment(),
17       this.todoSegment(),
18       this.footerSegment(),
19     ]);
20   }
21   ...
22 }
```

Isečak koda 4.4: Fajl `App.ts` nakon dodate funkcionalnosti

Ovaj C program se može prevesti pomoću prevodioca GCC [1].

Bibliografija

[1] Free Software Foundation. GNU gcc, 2013. on-line at: `http://gcc.gnu.org/`.

Biografija autora

Vuk Stefanović Karadžić (*Tršić, 26. oktobar/6. novembar 1787. — Beč, 7. februar 1864.*) bio je srpski filolog, reformator srpskog jezika, sakupljač narodnih umotvorina i pisac prvog rečnika srpskog jezika. Vuk je najznačajnija ličnost srpske književnosti prve polovine XIX veka. Stekao je i nekoliko počasnih doktorata. Učestvovao je u Prvom srpskom ustanku kao pisar i činovnik u Negotinskoj krajini, a nakon sloma ustanka preselio se u Beč, 1813. godine. Tu je upoznao Jerneja Kopitara, cenzora slovenskih knjiga, na čiji je podsticaj krenuo u prikupljanje srpskih narodnih pesama, reformu ćirilice i borbu za uvođenje narodnog jezika u srpsku književnost. Vukovim reformama u srpski jezik je uveden fonetski pravopis, a srpski jezik je potisnuo slavenosrpski jezik koji je u to vreme bio jezik obrazovanih ljudi. Tako se kao najvažnije godine Vukove reforme ističu 1818., 1836., 1839., 1847. i 1852.