

UNIVERZITET U BEOGRADU
MATEMATIČKI FAKULTET

Nemanja Subotić

IMPLEMENTACIJA PORTALA
METODOLOGIJA STRČNOG I NAUČNOG
RADA KORIŠĆENJEM FUNKCIJONALNIH
OKRUŽENJA ELM I PHOENIX

master rad

Beograd, 2020.

Mentor:

dr Milena VUJOŠEVIĆ JANIČIĆ, docent
Univerzitet u Beogradu, Matematički fakultet

Članovi komisije:

dr Ana ANIĆ, vanredni profesor
University of Disneyland, Nedodija

dr Laza LAZIĆ, docent
Univerzitet u Beogradu, Matematički fakultet

Datum odbrane: _____

Naslov master rada: Implementacija portala Metodologija strčnog i naučnog rada
korišćenjem funkcijonalnih okruženja Elm i Phoenix

Rezime: Apstrakt rada

Ključne reči: elm, elixir, ...

Sadržaj

1	Uvod	1
2	Elm	2
2.1	Uputstvo za instalaciju	3
2.2	Osnovne odlike	3
2.3	Elm kao platforma	4
2.4	Elm kao jezik	5
2.5	Elm arhitektura	6
3	Razvojno okruženje Phoenix i Elixir	8
4	Implementacija MSNR portala	9
5	Zaključak	10
	Bibliografija	11

Glava 1

Uvod

Funkcionalno programiranje kao programska paradigma nastaje 1959.godine sa pojavom LISP-a, prvog funkcionalnog programskog jezika... Elm... Phoenix i Elixir... MSNR Poral...

Glava 2

Elm

2012. godine Evan Zaplicki je objavio svoju tezu „Elm: Konkurento FRP ¹ za funkcionalne GUI-je ²” (eng. „Elm: Concurrent FRP for Functional GUIs”) [1] i, s ciljem da GUI programiranje učini prijatnijim, dizajnirao novi programski jezik - Elm. Elm je statički tipiziran, čisto funkcionalni programski jezik koji se kompilira, tačnije transpilira, u JavaScript i namenjen je isključivo za kreiranje korisničkog interfjesa veb aplikacija. Takođe, Elm nije samo programski jezik već i platforma



Slika 2.1: Logo Elm-a

za razvoj aplikacija. Zbog svoje funkcionalne prirode i prisustva kompilatora, Elm spada među najstabilnija i najpouzdanija razvojna okruženja, a za Elm aplikacije važi da, u praksi, ne izbacuju neplanirane greške tokom izvođenja (*eng. No Runtime Exceptions*).

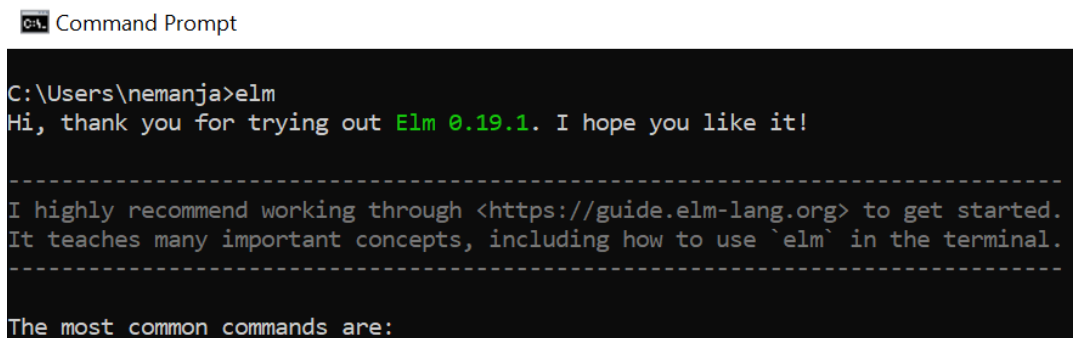
¹FRP-Funkcionalno Reaktivno Programiranje

²GUI - Grafički korisnički interfjes

2.1 Uputstvo za instalaciju

Pored želje da frontend programiranje učini prijatnijim, kreator jezika nastoji da ono bude i pristupačnije. Stoga, da biste počeli sa korišćenjem Elm-a instalacija nije potrebna, dovoljeno je otići na zvaničnu veb stranicu i pokrenuti online interaktivni kompilator [5], gde možete naći dosta primera, kao i vodić kroz Elm.

Za zahtevnije projekte neophodno je izvršiti instalaciju, koja je vrlo jednostavna. Potrebno je samo pratiti instrukcije sa zvanične stranice [3]. Provera uspešne instalacije može se izvršiti pokretanjem komande **elm** u komandnoj liniji, gde će se prikazati poruka dobrodošlice i spisak mogućih komandi o kojima će biti reč u sledećim poglavljima.



```
C:\Users\nemanja>elm
Hi, thank you for trying out Elm 0.19.1. I hope you like it!

-----
I highly recommend working through <https://guide.elm-lang.org> to get started.
It teaches many important concepts, including how to use `elm` in the terminal.
-----
The most common commands are:
```

Slika 2.2: Elm - Cmd

Takođe, moguća je instalacija pomoću **npm**³ alata [4].

2.2 Osnovne odlike

Pored *No Runtime Exceptions*, jedna od glavnih odlika ovog jezika jeste kompilator, koji je izuzetno ugodan za rad. Mnogi programeri smatraju da Elm kompilator proizvodi najbolje poruke o greškama. Za razliku od drugih, Elm kompilator objašnjava zašto je došlo do greške i daje predloge za njihovo rešavanje, a takođe nema kaskadnih poruka. Kreator se vodio razmišljanjem da kompilator treba da bude asistent, ne samo alat.

Elm koristi svoju verziju *virtualnog DOM-a*, koncepta koji se koristi u mnogim frontend okruženima. Ideja je da se u memoriji čuva „virtualna” reprezentacija

³Node Package Manager - JavaScript menadžer paketa

korisničkog interfejsa na osnovu koje se ažurira „stvarni” DOM. Još jeda bitna karakteristika Elm-a je nepromenljivost podataka, što znači da se jednom definisani podaci ne mogu više menjati. Direkta posledica nepromenljivost podataka je veoma brzo renderovanje HTML-a, jer se poređenja u virtuelnom DOM-u mogu vršiti po referenci. Verzija Elm 0.17 imala je najbrže renderovanje u poređenju sa tadašnjim aktuelnim verzijama popularnih okruženja.

Elm se može integrisati i u postojeće JavaScript projekte za implementaciju pojedinačnih komponenti. Takođe, moguća je i komunikacija između Elm-a i JavaScripta.

2.3 Elm kao platforma

Elm sa sobom donosi niz alata (tabela 2.1) i Elm okruženje (*eng. Elm Runtime*), koji su neophodni za razvoj i izvršavanje aplikacija. Elm kod se nalazi u datotekama sa *.elm* ekstenzijom i prilikom kompilacije kreira se jedna izlazna *.js* datoteka. U izlaznoj datoteci se pored prevedenog koda iz ulaznih *.elm* datoteka nalaze i (runtime) funkcije iz Elm okruženja potrebne za izvršavanje programa.

Alati	Kratat opis
repl	Pokretanje interaktivne sesije (<i>eng. Read-Eval-Print-Loop</i>)
init	Inicijalizacija projekta
reactor	Pokretanje lokalnog servera
make	Upotreba kompilatora
install	Preuzimanje paketa
diff	Prikazivanje razlika između različitih verzija istog paketa
bump	Određivanje broja naredne verzije paketa
publish	Publikacija paketa

Tabela 2.1: Elm alati komandne linije

Kao zaseban jezik Elm ima i zaseban sistem za upravljanje paketima. Pokretanjem komande **elm init** kreira se prazan *src* direktorijum i datoteka *elm.json*, u kojoj se pored informacije o tipu projekta (aplikacija ili paket), Elm verzije i liste direktorijuma sa kodom, nalazi i spisak paketa koji se koriste u projektu. Dodavanje novog paketa se vrši pomoću komande **elm install naziv-paketa**. Svi paketi nalaze se na <https://package.elm-lang.org/>, nazivi paketa su oblika *autor/ime-paketa*.

Kompilacija se vrši naredbom **elm make <jedna-ili-više-elm-datoteka>**, ukoliko se ne navede izlazna datoteka pomoću argumenta *--output* generisaće se *index.html*

datoteka sa prevedenim JavaScript kodom. Ostali argumenti ako i više informacija o drugim alatima može se videti pomoću naredbe **elm naziv-alata --help**

2.4 Elm kao jezik

„Rekao bih da je Elm ML sa sintaksom poput Haskell-a. Ako poredimo semantiku, Elm je dosta sličniji OCaml-u i SML-u.” –*Evan Czaplicki [2]*

ML (*eng. Meta Language*) je statički tipiziran programski jezik opšte namene koji je razio Robin Miler 1978. godine na Univerzitetu u Edinger. Nastao je pod uticajem LISP-a i ISWIM-a (*eng. If you See What I Mean*) jezika, pripada funkcionalnoj i imperativnoj paradigmi. Neke od osnovnih karakteristika jesu poklapanje obrazaca, Karijeve funkcije, poziv po vrednosti i posedovanje sakupljaca otpadaka. ML nije čist funkcionalan jezik i nema ugrađenu podršku za lenjo izračunavanje. U porodicu ML jezika, između ostalih, spadaju i **Standard ML**, **OCaml** i **F#**.

Haskell je čist funkcionalni programski jezik zasnovan na lambda računu. Naziv je dobio po matematičaru i logičaru Haskellu Bruks Kariju. Haskell je strogo tipiziran, poseduje automatsko zaključivanje tipova i lenjo izračunavanje. Jezik je opšte namene, pruža podršku za paralelno i distirbuirano programiranje. Haskell omogućava manje grešaka i veću pouzdanost kroz kraći, čistiji i održiviji kod.

Osnovni tipovi podataka

```
> 'Z'
'Z' : Char
> "Zdravo!"
"Zdravo!" : String
> True
True : Bool
> 42 / 10
4.2 : Float
> 42 // 10 --celobrojno deljenje
4 : Int
```

Listing 1: Osnovni tipovi podataka (elm repl)

Osnovni tipovi podataka u Elm-u su **Char**, **String**, **Bool**, **Int** i **Float**. U Listingu 1 prikazani su osnovni tipovi korišćenjem interpretera, koji nakon unetog izraza ispisuje izračunatu vrednost i tip.

Tip **Char** služi za predstavljanje unikod (*eng. unicode*) karaktera. Karakteri se navode između dva apostrofa ('a', '0', '\t'...), a moguće je koristiti i unikod zapis '\u{0000}' - '\u{10FFFF}'.

Za razliku od Haskell-a, gde je **String** zapravo **Char** lista, u Elm-u je poseban tip i predstavlja sekvencu unikod karaktera. Sekvenca se navodi između jednostrukih ili trostrukih navodnika.

```
> "\t String u jednom redu: escape navodnici \"Zdravo!\""
"\t String u jednom redu: escape navodnici \"Zdravo!\"" : String
>
> """String u više redova
  sa "navodnicima"! """
"String u više redova\n sa \"navodnicima\"! " : String
```

Listing 2: Stringovi

Bool predstavlja logički tip i može imati vrednost **True** ili **False**.

Int se koristi za prikazivanje celih brojeva. Siguran opseg vrednosti je od -2^{31} do $2^{31} - 1$, van toga sve zavisi od cilja kompilacije. Kada se prevodi u JavaScript, opseg se proširuje na -2^{53} do $2^{53} - 1$ u nekim operacijama, što ne bi važno ukoliko bi se, nekada kasnije, umesto JavaScript koda generisao WebAssembly, tada bi postojalo prekoračenje celih brojeva (*eng. integer overflow*). Vrednosti se mogu navoditi i u heksadecimalnom obliku (0x2A, -0x2b).

Float služi za predstavljanje brojeva u pokretnom zarezu po standardu *IEEE 754*. Vrednosti se mogu navoditi i pomoću eksponencijalnog zapisa, a decimalna tačka se mora nalaziti između dve cifre. Takođe u skup vrednosti spadaju NaN i Infinity.

U Listingu 3

Liste

2.5 Elm arhitektura

```
>0x2A
42 : number
> 1e3
1000 : Float
> 0/0
NaN : Float
> 1/0
Infinity : Float
```

Listing 3: Brojevi

Glava 3

Razvojno okruženje Phoenix i Elixir

Glava 4

Implementacija MSNR portala

Glava 5

Zaključak

Bibliografija

- [1] Evan Czaplicki. *Elm: Concurrent FRP for Functional GUIs*. 2012. URL: <https://elm-lang.org/assets/papers/concurrent-frp.pdf>.
- [2] Evan Czaplicki. *Github comment*. 2015. URL: <https://github.com/elm/elm-lang.org/issues/408#issuecomment-151656681>.
- [3] *Install Elm*. URL: <https://guide.elm-lang.org/install/elm.html>.
- [4] *Npm Elm*. URL: <https://www.npmjs.com/package/elm>.
- [5] *Try Elm*. URL: <https://elm-lang.org/try>.