Uvod u React.js

React.js je open-source JavaScript biblioteka napravljena za efikasnu izradu interaktivnih i kompleksnih korisničkih interfejsa. Originalno je kreirana od strane kompanije Meta (tadašnje kompanije Facebook) i od tada je stekla široku podršku zajednice veb programera zbog svog inovativnog pristupa manipulaciji DOM-a (Document Object Model), deklarativne paradigme i arhitekture zasnovane na komponentama.

Istorijat

2011. godine, Facebook je imao veliki broj korisnika (oko 845 miliona) i suočio se sa velikim izazovom. Kompanija je nastojala da Facebook platforma korisnicima ponudi bogatije i intuitivnije korisničko iskustvo kroz izradu bržeg i efikasnijeg korisničkog interfejsa.

Jedan od inženjera kompanije, Džordan Volk, je kreirao React upravo da bi rešio ranije pomenuti problem. React je pojednostavio proces izrade korisničkog interfejsa ponudom organizovanijeg i struktuiranijeg načina izrade dinamičkih interaktivnih veb stranica.

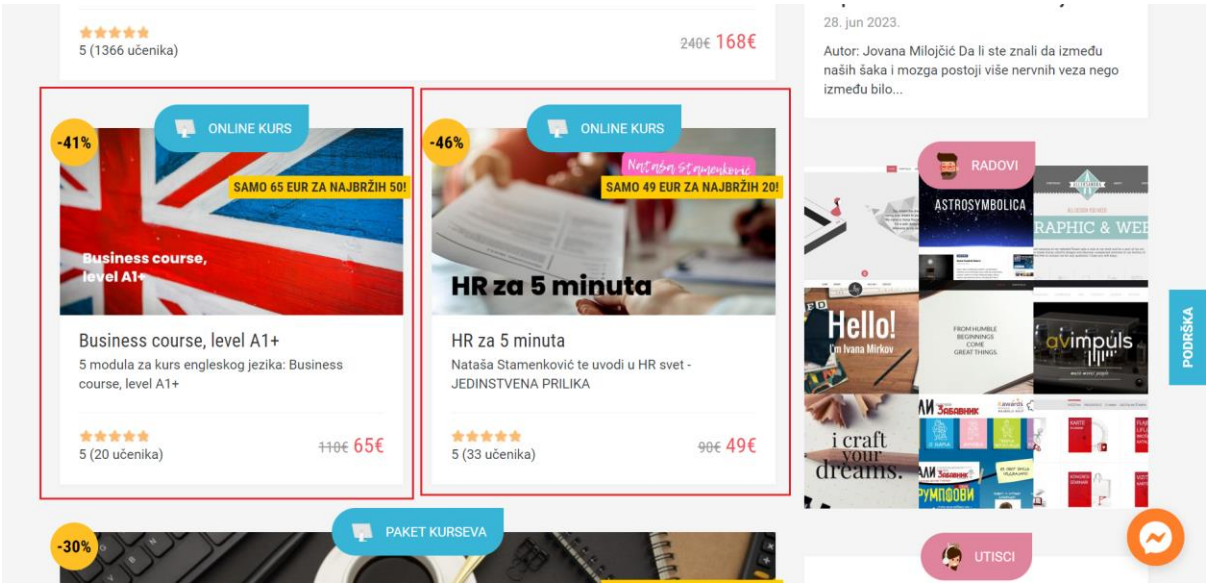
Kao projekat otvorenog koda je predstavljen široj javnosti 2013. godine. Od svog lansiranja, trajno je promenio pristup izradi veb aplikacija i ubrzo postao popularan u JavaScript ekosistemu.

Ključne karakteristike React.js biblioteke

Komponente

Komponente su osnovni i ključni koncept ove biblioteke. Predstavljaju ponovno upotrebljive i autonomne gradivne jedinice React.js aplikacije.

Umesto postojanja jednog fajla za definisanje strukture čitave veb stranice (.html), drugog za definisanje stila (.css) i trećeg za definisanje logike (.js), React ohrabruje potpuno drugačiju strukturu fajlova i pristup izradi stranice. U svetu React-a, veb stranica se odvaja u celine koje imaju samostalnu logiku i koje se mogu ponovo upotrebiti. Svaka od ovih komponenti se predstavlja zaseban fajl, u kom se, koristeći JSX sintaksu o kojoj će biti više reči kasnije, definišu struktura, stil i logika te komponente.



Na gornjoj slici se nalazi screenshot platforme Krojačeva škola. Obratimo pažnju na kartice istaknute crvenim okvirom. Zajedničke karakteristike su im:

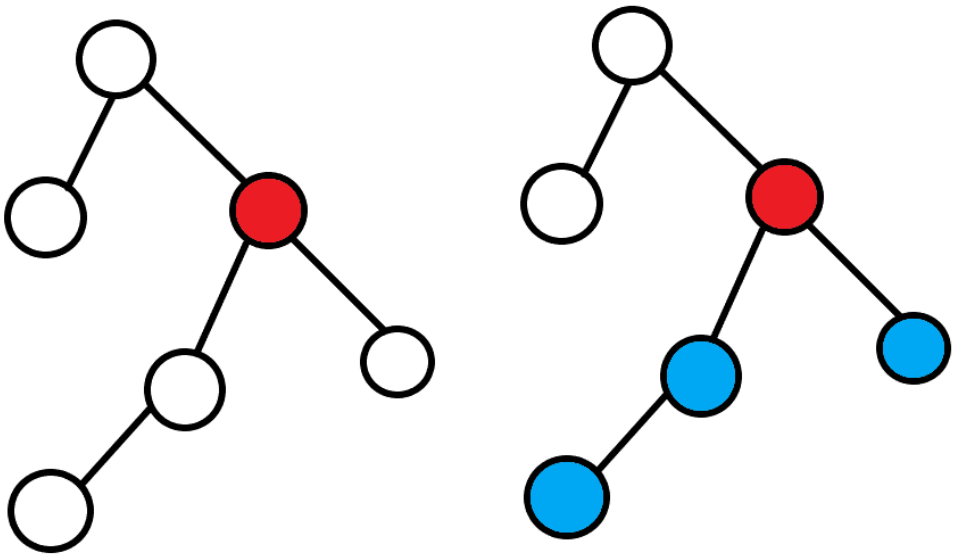
1. oznaka da se radi o online kursu
2. istaknut procenat popusta
3. oznaka “SAMO _ EUR ZA NAJBRŽIH _!”
4. baner (fotografija)
5. naslov kursa
6. opis kursa
7. ocena
8. precrtana prethodna i istaknuta nova cena
9. pritiskom na karticu se otvara puna naslovna stranica samog kursa

Korišćenjem React-a, mogli bismo da definišemo komponentu **KarticaKursa** koja bi predstavljala šablon ove kartice. Jednom kada to uradimo, na ovoj i bilo kojoj drugoj stranici ga možemo iskoristiti koliko god puta želimo. Prosleđivanjem različitih podataka svakoj od kartica možemo da dobijemo potpuno različite kartice izrađene po istom šablonu. Tako recimo, iako su obe od ovih kartica izrađene po šablonu **KarticaKursa**, jedna će pritiskom odvesti na stranicu kursa “Business course, level A1+”, dok će druga odvesti do stranice kursa “HR za 5 minuta”, samo zbog toga što su različitim instancama komponente **KarticaKursa** prosleđene različite URL adrese na koje treba da odvedu.

Ovaj pristup možemo primeniti i u izradi svih drugih elemenata veb stranice.

Jednosmerni tok podataka

Još jedan važan koncept React biblioteke predstavlja jednosmerni tok podataka, od roditelja ka deci.



Posmatrajmo ovaj dijagram i na trenutak se vratimo na koncept komponenti i zamislimo da je svaki od čvorova dijagrama jedna komponenta. Međusobni odnos komponenti je isti kao HTML elemenata. Jedna komponenta drugoj može biti **sibling** (na istom su nivou), **parent** (nivo iznad) i **child** (nivo ispod).

Osnovna ideja koncepta jednosmernog toka podataka jeste da ukoliko dođe do neke promene roditeljske komponente, te promene je svesna isključivo ona sama i “deca” te komponente, koja se takođe mogu promeniti u skladu sa promenama roditeljske komponente. U slučaju dijagrama, roditeljska komponenta je označena crvenom bojom, dok su “deca” koja snose posledice te promene označena plavom bojom.

Ključni razlozi zbog kojih se React zasniva na konceptu jednosmernog toka podataka su sledeći:

- **Predvidljivost**
  - Jednosmerni tok podataka omogućava jasan i predvidljiv tok podataka u aplikaciji.
  - Sve promene podataka je lako ispratiti i razumeti.
- **Otklanjanje grešaka**
  - S obzirom na to da je tok podataka u aplikaciji jednosmeran, mnogo je lakše identifikovati mesto greške i otkloniti je.
- **Čitljivost koda i njegova struktura**
  - Jednosmerni tok podataka kod čini čitljivijim, struktuiranijim i organizovanijim.
- **Optimizacija perfomansi**
  - Jednosmerni tok podataka omogućava React-u da efikasno izmeni virtuelni DOM i odredi minimalnu količinu modifikacija potrebnih za izmenu stvarnog DOM-a.
- **Olakšano testiranje**
  - Jednosmerni tok podataka omogućava testiranje individualnih komponenti sa vrlo konkretnim ulaznim podacima.
  - Ovo testiranje čini znatno modularnijim, zbog činjenice da se ponašanje svake komponente može testirati nezavisno.
- **Kompatibilnost sa funkcionalnim programiranjem**
  - Jednosmerni tok podataka je u kompatibilnosti sa principima funkcionalnog programiranja, akcentujući *nepromenljivost objekata* i *čiste funkcije*[1].

Virtuelni DOM i deklarativna paradigma

Paradigma programiranja na koju je većina početnika navikla jeste imperativna paradigma. Prateći ovu paradigmu, programer pisajući kod definiše algoritam i sve korake izvršavanja programa od unošenja ulaznih podataka, sve do krajnjeg rezultata.

Kada se vrši ručna manipulacija DOM-om veb stranice, primenjuje se upravo ova paradigma. Programer "uhvati" određeni HTML element koristeći *.querySelector()* ili *.getElementById(),* definiše događaj koji će prouzrokovati modifikaciju tog elementa, i zatim ga modifikuje logikom koju je sam definisao.

React koristi koncept deklarativne paradigme. Deklarativna paradigma je karakteristična po tome što se programer ne bavi definisanjem algoritma kojim će se izvršiti željena akcija. Programer se ovde bavi što preciznijim definisanjem rezultata te akcije, dok sam programski jezik i/ili bibiloteka sadrže ugrađene mehanizme koji služe za optimalno izvršavanje ove akcije.

Tu na scenu stupa virtuelni DOM. Virtuelni DOM je veliki JavaScript objekat koji predstavlja kopiju stvarnog DOM-a. Kucajući kod, React developer zapravo vrši modifikaciju virtuelnog DOM-a. Virtuelni DOM je doduše sinhronizovan sa stvarnim DOM-om, zbog čega, odmah nakon modifikacije virtuelnog DOM-a, nastupa modifikacija stvarnog DOM-a, ali koristeći optimizovane mehanizme same React biblioteke.

Na programeru je da opiše početno stanje aplikacije i načine na koje se ona menja, sve ostalo rešava sam React.

Ekosistem

Ekosistem ove biblioteke je ogroman. Trenutno je jedan je od najvećih u svetu JavaScript programskog jezika, a u njega su uključene druge biblioteke, alati i prakse koje dopunjuju

[1] čiste funkcije (pure functions) - funkcije koje za iste ulazne podatke uvek vraćaju identičan rezultat i ni na jedan način ne menjaju promenljive i objekte koji su postojali pre njihovog poziva

funckionalnosti React-a. Postoji kako bi pružio programerima optimalna rešenja za probleme sa kojima se susreću na svakodnevnoj bazi u radu sa ovom bibliotekom. Primarna uloga ekosistema React biblioteke se ogleda u proširivanju spektra mogućnosti ove biblioteke, pobošljanja održivosti koda i omogućavanja efikasnog skaliranja aplikacije.

Neki od esencijalnih elemenata React ekosistema su:

- **Za rutiranje:**
    - React Router
    - Next.js
- **Za state-management:**
    - Redux Toolkit
    - Zustand
- **Za testiranje:**
    - ViTest
    - React Testing Library
- **Za stilizovanje:**
    - Tailwind CSS
    - Styled Components
- **Za gotove komponente:**
    - Material-UI
    - Ant Design