SVEUČILIŠTE U SPLITU  
FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE

SEMINARSKI RAD

REACT

Mijo Jurić-Pešić, Roko Jaman

Marko Jokić

Split, travanj 2024.

Tablica 1‑1 Doprinosi koautora rada

|  |  |
| --- | --- |
| **Koautor** | **Kratki opis doprinosa koautora u izradi rada** |
| Mijo Jurić-Pešić | Odabir i obrada teme te njezinih podnaslova |
| Roko Jaman | Odabir i obrada teme te njezinih podnaslova |
| Marko Jokić | Odabir i obrada teme te njezinih podnaslova |

**IZJAVA**

Ovom izjavom potvrđujem da smo seminarski rad s naslovom „**React**“ pod mentorstvom prof. dr. sc. Ivana Nepostojećeg pisali samostalno, primijenivši znanja i vještine stečene tijekom studiranja na Fakultetu elektrotehnike, strojarstva i brodogradnje, kao i metodologiju znanstveno-istraživačkog rada te uz korištenje literature koja je navedena u radu. Spoznaje, stavove, zaključke, teorije i zakonitosti drugih autora koje smo izravno ili parafrazirajući naveli u završnom radu citirali smo i povezali s korištenim bibliografskim jedinicama.

Studenti

Mijo Jurić-Pešić

Roko Jaman

Marko Jokić

**SADRŽAJ**

[1 UVOD 1](#_Toc163670289)

[2 ŠTO JE REACT? 2](#_Toc163670290)

[2.1 Početak Reacta kao JavaScript biblioteke 2](#_Toc163670291)

[2.2 Jordan Walke- um iza Reacta 2](#_Toc163670292)

[3 UVOD U REACT 4](#_Toc163670293)

[3.1 Dinamičko ažuriranje korisničkog sučelja 4](#_Toc163670294)

[3.2 Upravljanje stanjem (State Management) 5](#_Toc163670295)

[3.3 Komponentni pristup (Component-based Approach) 6](#_Toc163670296)

[4 ŠTO ČINI REACT BOLJIM? 8](#_Toc163670297)

[4.1 Usporedba sa Next-JS-om 8](#_Toc163670298)

[4.1.1 Server-side rendering (SSR) 8](#_Toc163670299)

[4.1.2 Static site generation (SSG) 8](#_Toc163670300)

[4.1.3 Napredni sustav usmjeravanja 9](#_Toc163670301)

[4.1.4 Integracija s API-ima 9](#_Toc163670302)

[4.1.5 Zaključak 9](#_Toc163670303)

[4.2 Usporedba sa Angularom 9](#_Toc163670304)

[4.2.1 Integracija s TypeScriptom 9](#_Toc163670305)

[4.2.2 Arhitektura aplikacije 10](#_Toc163670306)

[4.2.3 Stvarni DOM vs. Virtualni DOM 10](#_Toc163670307)

[4.2.4 Zaključak 10](#_Toc163670308)

[5 GLAVNE ZNAČAJKE REACTA 12](#_Toc163670309)

[5.1 Alati 13](#_Toc163670310)

[5.2 Varijable 13](#_Toc163670311)

[5.3 Flux i Redux 13](#_Toc163670312)

[5.4 JSX (JavaScript i XML) 14](#_Toc163670313)

[6 NEDOSTATCI 15](#_Toc163670314)

[6.1 Strma krivulja učenja 15](#_Toc163670315)

[6.2 Složena konfiguracija 15](#_Toc163670316)

[6.3 Povezanost s drugim alatima 16](#_Toc163670317)

[6.4 Izazovi koje donosi SEO (Search Engine Optimization) 16](#_Toc163670318)

[7 ZAKLJUČAK 18](#_Toc163670319)

[LITERATURA 19](#_Toc163670320)

[PRILOZI 20](#_Toc163670321)

[Kazalo slika, tablica i kodova 20](#_Toc163670322)

[Kazalo slika 20](#_Toc163670323)

[Kazalo tablica 20](#_Toc163670324)

[Popis oznaka i kratica 20](#_Toc163670325)

[SAŽETAK I KLJUČNE RIJEČI 21](#_Toc163670326)

# UVOD

React je JavaScript biblioteka otvorenog kôda koju je razvio Facebook 2013. godine pod vodstvom Jordan Walkea da riješi probleme kod izrade korisničkih sučelja. Od danas je održavan od strane čitave zajednice React programera te se zbog toga brzo razvija i dobija nova rješenja i alate.

Glavne stavke koje čine React su: komponentni pristup kojim čitavo sučelje razdvoji na manje cjeline te na taj način čini kôd i sučelje čitljivijim i održivijim, virtualni DOM kojim kada se promjene dogode na pojedinoj komponenti ne ažurira čitavi DOM, već samo promijenjenu komponentu i upravljanje stanjem kojim smanjujemo vrijeme učitavanja i promet koji se odvija preko mreže te samim time smanjujemo i mogućnost da se dogodi greška.

Iako donosi niz poboljšanja, postoje i problemi , kojih ćemo se dotaknuti kasnije, a neki od njih su: strma krivulja učenja, složena konfiguracija, povezanost s drugim alatima te izazovi kod SEO optimizacije.

Glavni cilj ovog seminarskog rada jest prikazati što je te kako funkcionira React. Prvo poglavlje govori o tome što je React, njegov početak i razvoj u JavaScript biblioteku koju danas poznajemo. U drugom poglavlju ćemo dati uvod u React te koje probleme rješava. Treće poglavlje ćemo iskoristiti da pokažemo što React čini boljim od ostalih frameworka usporedivši ga sa Next JS-om te Angularom, a u četvrtom poglavlju ćemo obraditi glavne značajke Reacta. Peto poglavlje ćemo iskoristiti da navedemo neke od nedostatka koje React ima te ćemo na kraju dati zaključak i sažetak u istoimenim poglavljima.

# ŠTO JE REACT?

## Početak Reacta kao JavaScript biblioteke

React se prvi puta pojavio 2013. godine te od tada postaje alat za razvijanje modernih web aplikacija. Kao glavni stvaratelj Reacta ističe se Jordan Walke koji je vodio Facebook tim koji je radio na razvoju istog. Sama potreba za razvijanjem Reacta je bila ta što su ljudi iz Facebooka uvidjeli potrebu za razvijanjem alata koji bi im olakšao razvoj čitavog korisničkog sučelja. Od tada, React se koristi za jednu jednostavnu stvar, a to je da se koristi prilikom prikaza podataka u korisničkom sučelju.

Facebookov tim je uvidio da sučelja postaju sve kompleksnija, a samim time i teže održiva i manje responzivna krajnjem korisniku. React je nastao kao odgovor na sve ove probleme, koristeći komponentni pristup, JSX (JavaScript i XML), virtualni DOM (Document Object Model) i brojna druga rješenja, React postaje rješenje za korisnička sučelja.

Za razliku od drugih biblioteka i frameworka, sama suština Reacta ostaje ista kao i na početku; olakšavanje prikaza podataka u korisničkom sučelju. Bez obzira koliko aplikacija postane složenom, React daje jednostavan uvid u pojedine module te samim time olakšava svaki daljnji razvoj te održavanje iste.

## Jordan Walke- um iza Reacta

Jordan Walke je bio važan član Facebookovog tima od 2011. do 2021. godine. Svojim inovativnim radom je ostavio, možemo reći, neizbrisiv trag u Facebooku te u razvoju korisničkog sučelja. Najpoznatiji je po tome što je stvorio React, revolucionarnu JavaScript biblioteku koja je transformirala način na koji razvijamo korisnička sučelja na internetu. Kao ključna figura u razvoju Reacta, Walke je bio vođa tima programera koji su zajedno radili na ovom projektu.

Osim Reacta, Walke se može pohvaliti da je i autor ReasonML-a, programskog jezika koji je utemeljen na OCamilu. Važne karakteristike ReasonML-a su: jednostavnost, čitljivost kôda i efikasnost istog. ReasonML se koristi kao alternativa JavaScripta, ali ima jednu bitnu stavku koja ga povezuje sa njime, naime, ReasonML se kompajlira u JavaScriptu te je samim time i kompatibilan sa svim već napisanim JavaScript kôdom.

Izrazito je bitno naglasiti utjecaj samog Jordana Walkea i njegovih alata i danas u Facebooku; i React i ReasonML se i danas koriste u Facebooku u razvoju aplikacija. Te tehnologije su danas neizostavne komponente prilikom razvoja bilo koje web aplikacije sa dinamičkim korisničkim sučeljem.

# UVOD U REACT

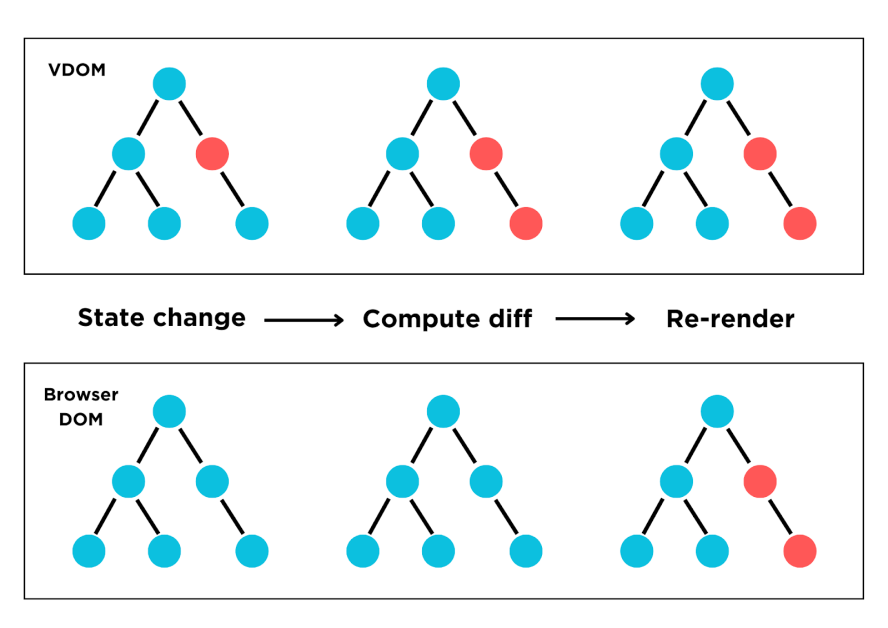
React je osmišljen da riješi probleme koje su donosile nove i složene web aplikacije. Utjecaj razvoja samog web-a se može najviše osjetiti na samim programerima; kako je rasla složenost istih, tako je opadala i produktivnost programera. S druge strane, korisnici su također bili ti koji su na drugoj strani osjetili značajan pad performansi na svojoj strani zbog povećane složenosti web aplikacija. „Zato se React razvio kao rješenje na specifičan set problema.“[1] Razmatrat ćemo tri glavna rješenja koja React nudi, a to su: dinamičko ažuriranje korisničkog sučelja (Dynamic UI Updates), upravljanje stanjem (State Managment) i komponentni pristup (Component-based Approach).

## Dinamičko ažuriranje korisničkog sučelja

„React koristi virtualni DOM te ponovno ažurira bilo što kada se stanje promjeni.“[2] React je poznat po svojoj efikasnoj upotrebi virtualnog DOM-a kako bi poboljšao performanse ažuriranja korisničkog sučelja. Kada dođe do promjene u stanju aplikacije, umjesto da odmah ažurira stvarni DOM, React prvo ažurira virtualni DOM. Ova tehnika omogućuje Reactu da izvrši samo minimalne promjene na stvarnom DOM-u, što rezultira bržim i efikasnijim ažuriranjem korisničkog sučelja.

Ključna prednost korištenja virtualnog DOM-a u Reactu je sposobnost djelomičnog ažuriranja. Umjesto da ažurira cijeli DOM svaki put kada se dogodi promjena, React uspoređuje virtualni DOM prije i nakon promjene. Zatim identificira samo dijelove koji su se promijenili i ažurira samo te dijelove na stvarnom DOM-u. Ova metoda djelomičnog ažuriranja znatno smanjuje opterećenje na pregledniku i ubrzava proces ažuriranja korisničkog sučelja

Korištenje virtualnog DOM-a je ključna karakteristika Reacta koja omogućuje efikasno ažuriranje korisničkog sučelja. Djelomično ažuriranje, kreiranje novog virtualnog prikaza i minimalne promjene na stvarnom DOM-u čine React snažnim alatom za razvoj performantnih web aplikacija. Kroz ovu inovativnu tehniku, React postiže bolje performanse u odnosu na tradicionalne pristupe ažuriranju DOM-a, čime osigurava fluidno korisničko iskustvo i visoku razinu efikasnosti u razvoju



Slika 3‑1 Usporedba virtualnog i stvarnog DOM-a

https://www.babbel.com/en/magazine/build-your-own-react-episode-2

## Upravljanje stanjem (State Management)

Kako web aplikacija raste, tako se povećava i složenost njezine strukture, što može dovesti do problema u održavanju i upravljanju kodom. U većim aplikacijama, posebno onima s više komponenti i složenijim stanjem, tradicionalni pristupi upravljanju stanjem mogu postati nepraktični i teški za održavanje. Međutim, React pruža elegantno rješenje ovog problema fokusirajući se isključivo na vizualni sloj aplikacije, dok ostavlja upravljanje stanjem aplikacije drugim alatima.

Korištenjem Reacta, programeri se usredotočuju samo na kreiranje komponenata i upravljanje prikazom podataka, dok se kompleksno upravljanje stanjem prepušta specijaliziranim bibliotekama i arhitekturama. Ovaj pristup omogućuje razdvajanje odgovornosti između različitih dijelova aplikacije, čime se olakšava razvoj, testiranje i održavanje koda.

Jedan od ključnih alata koji nadopunjuje React u upravljanju stanjem je Flux. Flux je jednosmjerni tok podataka koji pomaže u organizaciji i upravljanju stanjem aplikacije na jasan i predvidiv način. Iako Flux nije sam po sebi Framework, već konceptualni model, mnoge implementacije su razvijene kako bi ga podržale. Među njima, Redux se ističe kao jedna od najpopularnijih biblioteka za upravljanje stanjem u React aplikacijama.

Redux pruža jasnu arhitekturu za upravljanje stanjem, koja se temelji na centraliziranom skladištu podataka (store) i jasno definiranim akcijama koje izazivaju promjene u tom stanju. Ovaj pristup omogućuje lako praćenje i održavanje stanja aplikacije, te olakšava razvoj aplikacija koje se sastoje od velikog broja komponenata i složenih interakcija.

Kroz kombinaciju Reacta i Reduxa, programeri mogu razvijati skalabilne, pouzdane i lako održive web aplikacije, čak i u najkompleksnijim scenarijima. Ovaj spoj omogućuje programerima da se fokusiraju na kreiranje visoko kvalitetnih korisničkih sučelja, dok se upravljanje stanjem aplikacije povjerava Reduxu, osiguravajući tako stabilnost i efikasnost aplikacije kako ona raste i evoluira.

## Komponentni pristup (Component-based Approach)

Koristeći komponente iz prethodnih projekata ili već postojeća softverska rješenja, možemo značajno uštedjeti vrijeme, resurse i trud developera. Ovo je jedan od ključnih razloga zašto React preferira komponentni pristup u razvoju web aplikacija. Komponentni pristup omogućuje korištenje već gotovih i testiranih komponenti koje povećavaju modularnost i skalabilnost same aplikacije. Umjesto ponovnog izmišljanja tople vode, programeri mogu koristiti postojeće komponente kao građevne blokove za svoje aplikacije, čime se ubrzava razvojni proces i smanjuje rizik od grešaka.

Jedna od ključnih prednosti komponentnog pristupa je mogućnost razvoja aplikacije "dio po dio". Umjesto da se cijela aplikacija razvija odjednom, programeri mogu raditi na pojedinim komponentama neovisno jedna o drugoj. Ovo olakšava timski rad, omogućujući različitim članovima tima da istovremeno rade na različitim dijelovima aplikacije bez međusobnih preklapanja ili konflikata. Također, omogućuje brže iteracije i testiranje, jer se promjene mogu lako primijeniti samo na određene komponente, a ne na cijelu aplikaciju.

React dodatno olakšava rad s komponentama razbijajući korisničko sučelje na manje, modularne komponente. Ovo omogućuje programerima da se fokusiraju na razvoj specifičnih funkcionalnosti ili dijelova aplikacije, umjesto da se moraju baviti kompleksnim cijelim sučeljem odjednom. Svaka komponenta ima jasno definiranu ulogu i odgovornost, što olakšava razumijevanje koda i održavanje aplikacije.

Kroz kombinaciju komponentnog pristupa i modularnosti, React omogućuje programerima da razvijaju skalabilne, modularne i lako održive web aplikacije. Ovaj pristup olakšava timski rad, povećava produktivnost i smanjuje vrijeme potrebno za razvoj aplikacije. Kroz korištenje već postojećih komponenti i dijelova aplikacija, programeri mogu brže i efikasnije izgraditi visokokvalitetne web aplikacije koje zadovoljavaju potrebe korisnika i poslovanja.

A diagram of components of a component state

Description automatically generated

Slika 3‑2 Komponentni pristup

https://www.scholarhat.com/tutorial/react/react-js-components

# ŠTO ČINI REACT BOLJIM?

U web-development industriji, dinamičan rast tehnologija, frameworkova i alata neprekidno stvara bogatstvo mogućnosti za razvoj front-end aplikacija. Odaberemo li pogodan framework ili biblioteku za naš projekt, možemo značajno olakšati proces razvoja i poboljšati kvalitetu našeg softvera. JavaScript, kao temeljna tehnologija za front-end development, nudi širok spektar opcija poput: Vue, Ember, Backbone, jQuery, Svelte i mnogih drugih. Međutim, u ovom seminaru fokusirat ćemo se na usporedbu Reacta s dva od najpopularnija frameworka u ovom trenutku: Next.js i Angular. Naš cilj je detaljno analizirati performanse, održavanje i ključne značajke ovih frameworka kako bismo pružili smjernice za najbolji odabir ovisno o konkretnim situacijama i potrebama projekta.

## Usporedba sa Next-JS-om

Next.js, kao razvojni Framework koji nadopunjuje React biblioteku, izdvaja se svojim naprednim mogućnostima koje često nedostaju u osnovnom React ekosustavu. Ovdje ćemo detaljnije istražiti neke od ključnih značajki Next.js-a i kako one utječu na razvoj web aplikacija.

A screenshot of a phone

Description automatically generated

Slika 4‑1 Usporedba Angulara i Next-Js-a

https://mobisoftinfotech.com/resources/blog/next-js-vs-react/

### Server-side rendering (SSR)

Jedna od ključnih značajki Next.js-a je podrška za server-side rendering (SSR). Umjesto da se HTML dokument generira na klijentskoj strani (u pregledniku), Next.js omogućuje generiranje HTML-a na serveru prije nego što se pošalje klijentu. Ovo poboljšava performanse aplikacija jer korisnici brže vide sadržaj stranice, a također donosi dodatne koristi poput boljeg SEO-a jer tražilice mogu indeksirati sadržaj prije njegovog prikazivanja korisnicima.

### Static site generation (SSG)

Pored SSR-a, Next.js podržava i static site generation (SSG). Ova značajka omogućuje generiranje potpuno statičkih web stranica, što je izuzetno korisno za brzo učitavanje i skalabilnost web stranica. SSG omogućuje kreiranje optimiziranih stranica koje se mogu izrađivati unaprijed i distribuirati kao statički sadržaj, čime se smanjuje opterećenje servera i poboljšava korisničko iskustvo.

### Napredni sustav usmjeravanja

Next.js dolazi s naprednim sustavom usmjeravanja (routing) koji olakšava organizaciju i upravljanje rutama u aplikaciji. Ovo uključuje dinamičko generiranje ruta, podršku za parametre u rutama, automatsko preusmjeravanje i još mnogo toga. Napredno upravljanje rutama pomaže u održavanju jasne i organizirane strukture projekta, što je ključno za skalabilnost i održavanje.

### Integracija s API-ima

Next.js olakšava integraciju s vanjskim API-ima i servisima. Koristeći ugrađene alate i biblioteke, možete jednostavno komunicirati s API-ima i obraditi podatke u vašoj aplikaciji. Ovo je posebno korisno za aplikacije koje ovise o vanjskim podacima ili trebaju prilagodljivu komunikaciju s backend sustavom.

### Zaključak

Next.js donosi niz naprednih značajki koje nadopunjuju React biblioteku i čine ga snažnim alatom za razvoj web aplikacija. Od server-side renderinga i static site generationa do naprednog sustava rutiranja i integracije s API-ima, Next.js nudi mnogo više od osnovnog Reacta. Odluka između Reacta i Next.js-a ovisi o složenosti projekta i potrebama za naprednijim značajkama, ali Next.js pruža odlično rješenje za razvoj modernih web aplikacija.

## Usporedba sa Angularom

Angular je još jedan moćan framework za razvoj web aplikacija, koji se ističe svojom integracijom s TypeScriptom, supersetom JavaScripta. Razvijen od strane Google-a 2016. godine, Angular je evolucija njegovog prethodnika AngularJS-a i donosi brojne poboljšane značajke i alate za moderni web development.

A group of logos on a white background

Description automatically generated

Slika 4‑2 Tvrtke koje koriste Angular i React

https://www.simform.com/blog/angular-vs-react/

### Integracija s TypeScriptom

Jedna od najznačajnijih razlika između Angulara i Reacta je u tome što Angular koristi TypeScript, dok je React orijentiran prema JavaScriptu. TypeScript donosi tipizaciju i strožu statičku analizu koda, što može olakšati otkrivanje grešaka pri razvoju aplikacija. Ova integracija često rezultira čišćim i sigurnijim kodom, posebno u većim projektima.

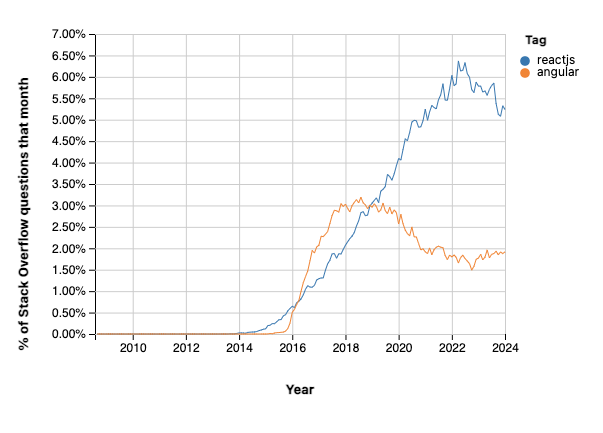
### Arhitektura aplikacije

Angular se temelji na arhitekturi Model-View-Controller (MVC), dok React preferira komponentni pristup. U Angularu, aplikacija se dijeli na tri glavna dijela: modele (Model), koji su odgovorni za pohranu podataka, prikaze (View), koji prikazuju podatke iz modela korisnicima, i kontrolere (Controller), koji upravljaju interakcijom između modela i prikaza. Ova jasna podjela uloga olakšava organizaciju i održavanje aplikacije, posebno u većim projektima.

### Stvarni DOM vs. Virtualni DOM

Jedna od bitnih razlika između Angulara i Reacta je u tome kako svaki od ovih alata upravlja DOM-om. Angular koristi stvarni DOM, dok React koristi virtualni DOM. Virtualni DOM omogućuje brže ažuriranje korisničkog sučelja, jer umjesto direktnog mijenjanja stvarnog DOM-a, React generira virtualnu reprezentaciju i uspoređuje je s prethodnom verzijom kako bi utvrdio potrebne promjene. Ovo može rezultirati boljim performansama aplikacije, posebno u slučajevima gdje postoji veliki broj komponenata koje se često mijenjaju.

### Zaključak

Odabir između Angulara i Reacta ovisi o specifičnim potrebama projekta, kao i preferencijama i iskustvu razvojnog tima. Angular je idealan za veće projekte koji zahtijevaju strožu organizaciju i arhitekturu, te integraciju s TypeScriptom. Sa svojom MVC arhitekturom i stvarnim DOM-om, Angular nudi snažan alat za razvoj kompleksnih web aplikacija. S druge strane, React je popularan zbog svoje jednostavnosti, virtualnog DOM-a i fleksibilnosti, što ga čini dobrom opcijom za manje projekte ili one koji zahtijevaju veću fleksibilnost u dizajnu i arhitekturi aplikacije. Važno je pažljivo razmotriti prednosti i nedostatke svakog alata kako bi se odabrala najbolja opcija za određenu situaciju.

Slika 4‑3 Razlika pretraživanja Reacta i Angulara na Stack Overflowu

https://dev.to/pagepro\_agency/react-vs-angular-which-one-to-choose-in-2021-1ddo

# GLAVNE ZNAČAJKE REACTA

React donosi niz ključnih značajki koje olakšavaju razvoj modernih web aplikacija. Sa svojom arhitekturom koja se temelji na komponentama, programeri mogu stvoriti elemente korisničkog sučelja poput gumba ili traka za pretraživanje, koji se zatim mogu ponovno koristiti u cijeloj aplikaciji. Ovo ne samo da usmjerava razvojni proces, već i poboljšava mogućnost održavanja. Jedna od prepoznatljivih značajki Reacta je JSX (JavaScript XML). JSX izgleda slično HTML-u i omogućuje razvojnim programerima pisanje oznaka izravno unutar svog JavaScript koda. Ova jedinstvena mogućnost pojednostavljuje kodiranje pružajući intuitivniji način strukturiranja komponenti. Reactov utjecaj ne završava na pregledniku; također igra vitalnu ulogu u razvoju mobilnih aplikacija s varijantom koja se zove React Native. React također koristi virtualni DOM (Document Object Model), koncept programiranja gdje se idealna ili "virtualna" reprezentacija korisničkog sučelja čuva u memoriji i sinkronizira sa "pravim" DOM-om Reactovim postupkom usklađivanja. Ova značajka donosi visoku učinkovitost u prikazivanju web stranica, čineći aplikacije bržima i osjetljivijima.

Upravljanje stanjem u Reactu je fleksibilno; dok postoji ugrađena podrška kroz 'state' u komponentama klase ili 'useState' kuku u funkcionalnim komponentama, React također prihvaća biblioteke trećih strana kao što je Redux za veće aplikacije. Kada odete na web lokaciju koristeći React, web server će vam poslati dvije stvari: veliku JavaScript datoteku, koja se obično naziva bundle.js; i šablonsku HTML stranicu koja učitava tu JS datoteku s oznakom <script>. Ovo čini učitavanje prve stranice sporijim s React aplikacijama, ali to nadoknađuje uvelike povećanjem narednih učitavanja stranica. Bundle datoteka sadrži sav kod koji čini da React radi zajedno sa svim kodom koji sama aplikacija koristi. Sve zavisnosti od npm također su grupirane u ovaj file.

Konačno, osim što je izvrstan za izgradnju korisničkih sučelja na web platformi sa svojom glavnom bibliotekom react-dom, također podržava razvoj mobilne platforme koristeći react-native. To omogućuje stvaranje i web-aplikacija i izvornih mobilnih aplikacija koristeći uglavnom slične obrasce kodiranja.

## Alati

React dolazi s bogatim skupom alata koji pomažu razvojnim timovima u izgradnji performantnih aplikacija. Budući da je najpopularniji front-end JavaScript okvir, React ekosistem je živ i pun korisnih alata i paketa koje možete integrirati u svoju aplikaciju Kao što je istaknuto, "React Developer Tools pruža duboke uvide u stanje vaše aplikacije i omogućuje vam učinkovito dijagnosticiranje performansi."[3] Ovaj alat omogućuje programerima da duboko pregledaju i razumiju stanje komponenti u stvarnom vremenu, poboljšavajući tako proces razvoja.

## Varijable

Varijable u Reactu omogućuju dinamičko upravljanje podacima i stanjem aplikacije. Kroz korištenje varijabli, "React pruža jednostavan način za upravljanje stanjem komponenti."[3] Ova fleksibilnost omogućuje programerima da lako manipuliraju podacima unutar komponenata, što dovodi do bržeg i efikasnijeg razvoja aplikacija.

## Flux i Redux

Flux i Redux su arhitekturni obrasci koji se često koriste s Reactom za upravljanje stanjem aplikacije. Kroz korištenje ovih biblioteka, "React aplikacije postaju skalabilnije i lakše za održavanje."[2] Ovi alati omogućuju efikasno upravljanje složenim stanjima aplikacije, što rezultira boljom skalabilnošću i održivošću projekta.

Redux je biblioteka otvorenog koda objavljena nekoliko godina nakon Reacta koja koristi koncept "pohrane" podataka kako bi pomogla u praćenju stanja aplikacije. Iako u Reactu možete dijeliti podatke o stanju među komponentama, ponekad je to teško učiniti učinkovito, posebno za složene aplikacije. U Reduxu sve komponente imaju izravan pristup istim trgovinama, što olakšava prijenos stanja između različitih dijelova aplikacije. Reduxovi alati za otklanjanje grešaka su fantastični. Budući da je stanje nepromjenjivo i može se ažurirati samo kroz akcije i reduktore, ono omogućava otklanjanje grešaka u vremenskom putu. Možete se pomicati unatrag i gledati kako vaše radnje utječu na trenutno stanje, te istražiti stablo stanja koristeći Redux Devtools.

Flux je aplikacijska arhitektura koju Facebook interno koristi za izgradnju klijentske web aplikacije s Reactom. Korisno je kada projekt ima dinamičke podatke, a moramo ažurirati podatke na učinkovit način. Smanjuje pogreške tijekom izvođenja.

A diagram of a diagram

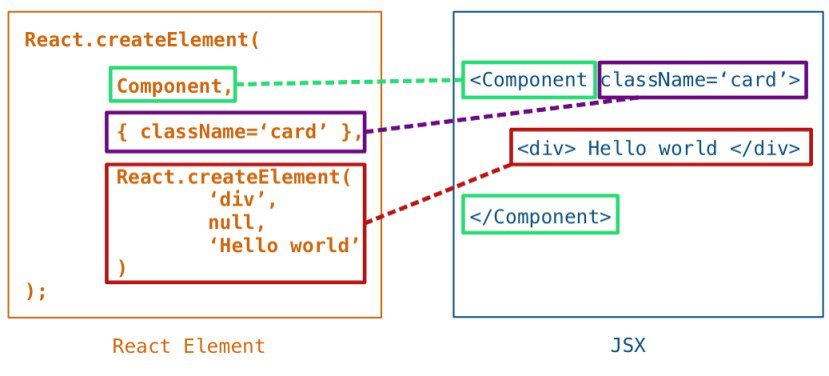
Description automatically generated with medium confidence

Slika 5‑1 Prednosti i nedostatci Fluxa i Reduxa

https://stackoverflow.com/questions/32761316/flux-vs-redux-pros-and-cons-highlights

## JSX (JavaScript i XML)

JSX (JavaScript XML, formalno JavaScript Syntax eXtension) je proširenje JavaScripta koje omogućuje stvaranje stabala Document Object Model (DOM) korištenjem sintakse slične XML-u.[1] U početku kreiran od strane Facebooka za korištenje s Reactom, JSX je usvojen od strane više web okvira. Umjesto umjetnog odvajanja tehnologija stavljanjem oznake i logike u zasebne datoteke, React odvaja probleme s fleksibilno povezanim jedinicama koje se nazivaju "komponente" koje sadrže oboje. React ne zahtijeva korištenje JSX-a, ali većina ga ljudi smatra korisnim kao vizualna pomoć pri radu s korisničkim sučeljem unutar JavaScript koda. Također omogućuje Reactu da prikaže korisnije poruke o pogrešci i upozorenja.



Slika 5‑2 Integracija JSX-a u React

https://www.geeksforgeeks.org/react-jsx-in-depth/

Svrha ove specifikacije je definirati sažetu i poznatu sintaksu za definiranje struktura stabla s atributima. Generička, ali dobro definirana sintaksa omogućuje zajednici neovisnih analizatora i označivača sintakse da se prilagode jednoj specifikaciji. Ugrađivanje nove sintakse u postojeći jezik riskantan je pothvat. Drugi implementatori sintakse ili postojeći jezik mogu uvesti drugo nekompatibilno proširenje sintakse. Ova specifikacija ne pokušava biti u skladu s XML ili HTML specifikacijom. JSX je dizajniran kao značajka ECMAScripta, a sličnost s XML-om služi samo za upoznavanje.

# NEDOSTATCI

Iako je React izuzetno popularan alat za izradu korisničkih sučelja, suočava se s nekoliko nedostataka. Strma krivulja učenja, koja uključuje koncepte kao što su virtualni DOM i JSX sintaksa, može biti izazovna za početnike ili developere koji se po prvi puta susreću s istom. Integracija s drugim alatima i bibliotekama može rezultirati složenom konfiguracijom i zahtijevati dodatno vrijeme i trud kako bi se iskoristio puni potencijal Reacta. Pored toga, izazovi vezani uz SEO (Search Engine Optimization) mogu se pojaviti kada se koristi React za izgradnju web stranica, jer dinamički generirani sadržaj može biti teže indeksiran od strane pretraživača. Unatoč tim izazovima, pravilno planiranje i dobra koordinacija mogu pomoći timovima da prevladaju ove izazove i stvore visokokvalitetne aplikacije pomoću Reacta.

## Strma krivulja učenja

Jedan od glavnih nedostataka Reacta je strma krivulja učenja koja se može pokazati kao prepreka, kako za iskusne, tako i za programere koji se po prvi puta susreću s njim. Koncepti koji „zakompliciraju“ učenje Reacta su: virtualni DOM, JSX te upravljanje stanjem kroz Redux. „JSX (JavaScript i XML) je sintaksni dodatak JaviScriptu koji nam omogućava da napravimo elemente Reacta sa sintaksom HTML-a unutar JavaScrpta.“[4] Treba vremena i predanosti kako bi se programeri potpuno osposobili za rad s ovim alatom. Dok je React poznat po svojoj jednostavnosti i efikasnosti, razumijevanje njegovih osnovnih koncepta može zahtijevati neko vrijeme i praksu.

Svi su ovi problemi rješivi na način da se JSX ne koristi. Iako će nam JSX uvelike olakšati razvoj čitavog korisničkog sučelja, sve što se napravi sa JSX-om može se napraviti i u samome JavaScriptu, ako konfiguracija to dopušta.

## Složena konfiguracija

Prilikom izrade web aplikacija s Reactom, često se javlja potreba za integracijom s drugim alatima i bibliotekama kako bi se postigla određena funkcionalnost ili poboljšala produktivnost razvoja.

Kada promatramo globalno upravljanje aplikacijom, možemo za to upotrijebiti Redux. Ali, sama konfiguracija čitavog Reduxa može iziskivati dodatno vrijeme i trud kojega developeri često nemaju. Isto tako kada korisnik želi preko URL-a pregledavati različite dijelove aplikacije, morat će koristiti React Router. Međutim, njegova je integracija u već postojeći React kôd izuzetno teška jer treba definirati rute te osigurati da aplikacija ispravno detektira promjene rute i slično.

Ovi se problemi rješavaju kroz praksu, pravilnu edukaciju i obuku razvojnog tima, korištenje dokumentacije te korištenjem već gotovih predložaka . Samim time, ekipe dobivaju uvid u funkcionalnost same aplikacije te produbljuju vlastita znanja o istome.

## Povezanost s drugim alatima

Razvoj web-aplikacija ponekad zahtijeva više od osnovne funkcionalnosti koju pruža React. U takvim slučajevima, razvojni timovi često moraju integrirati React s dodatnim alatima kako bi postigli željenu funkcionalnost i poboljšali korisničko iskustvo.

Primjerice, koristeći već spomenuti Redux olakšat ćemo globalno upravljanje aplikacijom te ćemo olakšati pristup pojedinoj komponenti što je posebno bitno kod složenijih aplikacija. Također, ako se naša aplikacija sastoji od više stranica ili je potrebno dodatno rješiti problem ruta, React nema tu mogućnost osim ako mu ne integriramo React Routing. Korištenjem njega, korisniku ćemo omogućiti da pregledavanje stranice i interakciju sa istom bez potrebe da se čitava stranica učitava. Takva ovisnost Reacta o drugim bibliotekama može učiniti React odbojnim novim korisnicima.

Kao i za sve drugo, i za ovo postoji rješenje, kao što je pravilno planiranje aplikacije (kako bismo znali koje biblioteke koristiti), pravovremena integracija biblioteka te korištenje ažurne dokumentacije.

## Izazovi koje donosi SEO (Search Engine Optimization)

Razvoj web stranica s Reactom donosi specifične izazove vezane uz optimizaciju za pretraživače (SEO), posebno kada se radi o statičkim web stranicama. Dok React omogućuje izgradnju dinamičkih korisničkih sučelja, često se koristi za izgradnju jednostraničnih aplikacija ili SPA (Single Page Application) koje generiraju dinamički sadržaj na temelju korisničkih interakcija.

Uzmimo za primjer Google; Google preferira statički HTML koji je lako indeksirati i prikazati na korisničkom sučelju. To su stranice kao što su: blogovi, stranice za kontakt ili podršku, stranice sa informacijama o događajima i slično. Dinamički generirani sadržaj koji se stvara pomoću JavaScripta, kao što je slučaj s Reactom, može otežati indeksiranje pretraživačima.

Kako bismo prevladali ove probleme, potrebno je koristiti SSR (Server Side Rendering) koji se koristi za renderenje stranice na serveru prije no što je isporučimo korisniku i pravilna upotreba meta tagova kojim ćemo postići da se aplikacija uspješno razvije.

A diagram of a search engine optimization process

Description automatically generated

Slika 6‑1 Procesi u Search Engine Optimizaciji

https://www.researchgate.net/figure/Search-engine-optimization-process\_fig1\_235763091

# ZAKLJUČAK

Ovaj seminarski rad donosi dublji uvid u React, ističući njegove ključne značajke, prednosti, ali i nedostatke. Kroz pregled početaka Reacta, ulogu njegovog osnivača Jordana Walkea, te uvod u njegovu funkcionalnost, seminarski rad pruža sveobuhvatan pregled ove popularne JavaScript biblioteke. Nadalje, kroz analizu tri glavna rješenja koja React nudi: dinamičko ažuriranje korisničkog sučelja, upravljanje stanjem i komponentni pristup - raspravljeno je kako React olakšava razvoj modernih web aplikacija i poboljšava korisničko iskustvo. U provedneoj usporedbi s drugim popularnim alatima poput Next.js-a i Angulara nudi se uvid u razlike i sličnosti te pomažući u donošenju informirane odluke o odabiru tehnologije ovisno o konkretnim potrebama projekta.

Kroz dublje istraživanje glavnih značajki Reacta poput JSX-a, virtualnog DOM-a, upravljanja stanjem te podrške za mobilni razvoj kroz React Native, zaključuje se da React pruža moćan alat za izgradnju visokokvalitetnih web i mobilnih aplikacija. Kroz pregled alata poput React Developer Tools, naglašava se podrška razvojnom timu i olakšavanje dijagnostike performansi, dok se kroz upravljanje varijablama, Flux i Redux ističu donoseći fleksibilnost i skalabilnost.

U konačnici, seminarski rad pruža uvid u širinu React ekosustava, ističući njegovu popularnost, ali i mane poput složene konfiguracije i složenog procesa učenja. Unatoč nedostatcima, React ostaje ključan alat u nizu modernih web alata i frameworka, kontinuirano poboljšavajući korisnička iskustva i omogućujući razvoj skalabilnih aplikacija s visokim performansama.

# LITERATURA

1. Gackenheimer, Cory. *Introduction to React*. Apress, 2015.
2. Chen, Songtao, Upendar Rao Thaduri, and Venkata Koteswara Rao Ballamudi. ‘Front-End Development in React: An Overview’. *Engineering International* 7, no. 2 (31 December 2019): 117–26. <https://doi.org/10.18034/ei.v7i2.662>.
3. ‘Facebookarchive/Flux’. JavaScript. 2014. Reprint, Meta Archive, 8 April 2024. <https://github.com/facebookarchive/flux>.
4. ‘React’. Accessed 7 April 2024. <https://react.dev/>.
5. Sofela, Oluwatobi. React Explained Clearly: All You Need to Build Great React.Js Apps. CodeSweetly, 2021.

# PRILOZI

## Kazalo slika, tablica i kodova

### Kazalo slika

[Slika 3‑1 Usporedba virtualnog i stvarnog DOM-a 5](#_Toc165972211)

[Slika 3‑2 Komponentni pristup 7](#_Toc165972212)

[Slika 4‑1 Usporedba Angulara i Next-Js-a 9](#_Toc165972213)

[Slika 4‑2 Tvrtke koje koriste Angular i React 11](#_Toc165972214)

[Slika 4‑3 Razlika pretraživanja Reacta i Angulara na Stack Overflowu 13](#_Toc165972215)

[Slika 5‑1 Prednosti i nedostatci Fluxa i Reduxa 16](#_Toc165972216)

[Slika 5‑2 Integracija JSX-a u React 17](#_Toc165972217)

[Slika 6‑1 Procesi u Search Engine Optimizaciji 20](#_Toc165972218)

### Kazalo tablica

[Tablica 1‑1 Doprinosi koautora rada 1](#_Toc163669984)

## Popis oznaka i kratica

DOM Document Object Model

JSX JavaScrpit i XML

XML Extensible Markup Language

SEO Search Engine Optimization

# SAŽETAK I KLJUČNE RIJEČI

**Sažetak**

React je JavaScript biblioteka koju je Facebook stvorio 2013. godine za razvoj modernih korisničkih sučelja. Temelji se na komponentnom pristupu, omogućujući programerima da podijele sučelje na manje, lako održive dijelove. Virtualni DOM omogućuje učinkovito ažuriranje promijenjenih dijelova, dok upravljanje stanjem pojednostavljuje složene interakcije. Unatoč popularnosti, React ima izazove, uključujući strmu krivulju učenja, složenu konfiguraciju i poteškoće s optimizacijom za pretraživače (SEO). Next.js proširuje mogućnosti Reacta s funkcijama poput server-side renderinga (SSR) i static site generationa (SSG). Angular, framework temeljen na TypeScriptu, koristi strogu arhitekturu Model-View-Controller (MVC). Redux je popularan alat za upravljanje stanjem u React aplikacijama. React također ima jedinstvene značajke poput JSX-a, što olakšava kombiniranje HTML-a i JavaScript koda. Iako nosi izazove, pravilno planiranje i integracija dodatnih alata mogu pomoći u prevladavanju potencijalnih problema.

**Ključne riječi**

React, Jordan Walke, Virtualni Dom, komponentni prstup, API, Angular, Next-JS, Flux, Redux, JSX