SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Pregled modernih web frontend radnih okvira

Marina Rupe
Voditelj: *Mile Šikić*

Zagreb, svibanj 2018.

SADRŽAJ

1. Uvod

Od kada je nastao pa do danas, internet je prošao veliku preobrazbu. Od male mreže koju je činilo tek četiri čvora do mreže od nekoliko milijardi korisnika diljem svijeta. Rastom broja korisnika, internet je mijenjao svoje obličje kako bi zadovoljio njihove potrebe.

Svaka web stranica predstavljena je strukturom koja se zove DOM (engl. *Document Object Model*). DOM je zapisan u obliku HTML (engl. *HyperText Markup Language*) oznaka (engl. *tagova*) koje objašnjavaju koja će biti struktura web stranice, odnosno koje elemente ona sadrži. HTML je nastao zbog želje za organizacijom sadržaja na web stranicama. U početku su to uglavnom bili znanstveni radovi koji su svi trebali slijediti sličnu strukturu i međusobno se referencirati. S pomoću HTML-a, mogle su se jasno odvojiti cjeline bilo kojeg dokumenta i međusobno ih povezati hipervezama. Uskoro se javila potreba za uređivanjem samog izgleda web stranice te je osmišljen CSS (engl. *Cascading Style Sheets*) koji služi za stiliziranje HTML elemenata.

U početku su na internetu prevladavale statičke web stranice koje su činili samo HTML i CSS, no ideje za dizajn stranica postale su sve kompliciranije i bilo je potrebno napraviti skriptni jezik koji bi učinio da se elementi stranice dinamički mijenjaju, uglavnom s ciljem postizanja zanimljivijeg i ljepšeg dizajna. Tako je 1995. godine za tu svrhu osmišljen programski jezik JavaScript [1]. Nekoliko je godina bilo dovoljno koristiti JavaScript samo kao skriptni jezik, kao dodatak web stranicama za neke specijalne efekte. Danas to više nije tako. Statičke web stranice sve se više zamjenjuju dinamičkim web aplikacijama koje imaju vrlo razvijeno sučelje, mogu manipulirati podacima, upravljati stanjem aplikacije, dinamički mijenjati prikazan sadržaj i sl. Kako bi sve to bilo moguće, bilo je potrebno osmisliti radne okvire koji bi pojednostavili razvoj takvih aplikacija i omogućili postojanje tzv. „debelih klijenata“, odnosno prebacivanja dijela aplikacijske logike s poslužiteljske strane na klijenta. Postoji mnogo biblioteka i radnih okvira za jezik JavaScript, a u ovom seminarskom radu obradit će se trenutno najpopularniji: Angular, React i Vue.

2. Osnovni koncepti

U ovom poglavlju obradit će se neki od osnovnih koncepata i pojmova na kojima se zasniva razvoj modernih klijentskih web aplikacija i kao takvi prisutni su i u radnim okvirima koji će se analizirati.

2.1. Aplikacija jedne stranice

Aplikacija jedne stranice ili SPA (engl. *Single Page Application*) je vrsta web aplikacije čija se komunikacija zasniva na ponovnom iscrtavanju elemenata trenutno učitane stranice, čime se izbjegava učitavanje cijelih novih stranica s poslužitelja. Dakle, iscrtavanje se obavlja na klijentskoj (engl. *client-side*) umjesto na poslužiteljskoj strani (engl. *server-side*). Na ovaj se način aplikacija ponaša sličnije *desktop* aplikacijama. Aplikacije jedne stranice idealno su rješenje za ostvarenje vrlo dinamičkih web aplikacija. Korisničko je iskustvo puno bolje jer korisnik ne mora čekati da mu se dohvati nova stranica s poslužitelja. Unatoč tome, aplikacija se ponaša kao uobičajena web aplikacija u smislu da i dalje postoji navigacija između stranica i pripadne rute, tj. URL-ovi (iako se pri odlasku na novu rutu stranica iscrtava na klijentskoj strani, a ne dohvaća se s poslužitelja). U aplikacijama jedne stranice, cijeli potreban kod se ili dohvati pri prvom učitavanju stranice ili se dinamički dohvaća po potrebi (najčešće kao odgovor na korisničke akcije). Većina modernih web radnih okvira omogućava razvoj aplikacija jedne stranice [2].

2.2. Web komponente

Web komponente (engl. *Web components*) su tehnologija koja omogućava kreiranje vlastitih HTML elemenata koji se mogu koristiti pri razvoju web aplikacija. Ideja web komponenti je enkapsulirati HTML kod, skripte i stil u nove elemente koji se kasnije mogu jednostavno ponovno iskoristiti.

Web komponente zasnivaju se na četiri koncepta: prilagođeni elementi (engl. *Custom Elements*), skriveni DOM (engl. *Shadow DOM*), HTML predlošci (engl. *HTML Templates*) i HTML učitavanja (engl. *HTML Imports*).

Prilagođeni elementi su skup JavaScript sučelja za definiranje novih HTML elemenata i njihovih ponašanja. Ti se elementi mogu kasnije koristiti za razvoj korisničkog sučelja.

Skriveni DOM je skup JavaScript sučelja za uključivanje i upravljanje enkapsuliranim „skrivenim“ stablom elemenata koji se iscrtavaju zasebno od ostatka DOM-a. Na ovaj se način mogu sakriti unutarnje funkcionalnosti elemenata i sprječavaju se kolizije među elementima (primjerice u stiliziranju ili skriptiranju elementa).

HTML predlošci uvode <*template*> i <*slot*> elemente koji omogućavaju pisanje predložaka s oznakama (engl. *markup template*) koji se ne prikazuju na iscrtanoj stranici. Ti se predlošci mogu kasnije višestruko iskoristiti uključivanjem u prilagođene elemente.

HTML učitavanja su mehanizam koji omogućava uključivanje kreiranih web komponenti u druge stranice. Cilj je razdvojiti definicije komponenti u zasebnu datoteku i potom koristiti komponente tako da ih se prije korištenja uključi u stranicu koristeći navedeni mehanizam [3].

Ovi se koncepti mogu implementirati djelomično ili u potpunosti, no nisu svi jednako podržani u svim web preglednicima [4]. Web komponente mogu se kao tehnologija koristiti neovisno o radnom okviru, no mnogi radni okviri iskoristili su ideju podjele aplikacije u zasebne komponente radi bolje organizacije koda i ponovnog iskorištavanja.

2.3. „Podaci dolje, akcije gore“ princip

„Podaci dolje, akcije gore“ (engl. *data down, actions up*) programski je princip koji se koristi u različitim radnim okvirima i na neki je način postao standard za moderne web radne okvire. Ideja ovog principa je da se podaci proslijeđuju iz viših prema nižim komponentama hijerarhije, dakle iz roditeljskih komponenti prema komponentama djeci kojima su potrebni podaci iz roditelja. Roditeljska komponenta u tom slučaju čuva podatke, a djeca te podatke samo pasivno koriste. U slučaju da se dogodi neka akcija u komponenti djetetu (npr. klik na gumb), kompenta dijete zadužena je tu akciju proslijediti do roditelja (smjer prema gore). Komponenta roditelj reagira na akciju i izmjenjuje podatke na odgovarajući način. Djeca nemaju uvid u način izmjene podataka, no podaci koje dobivaju od roditeljske komponente će se na ovaj način ažurirati. Na

ovaj se način ostvaruje učinkovit protok podataka unutar aplikacije i pojednostavljuje komunikaciju među komponentama, posebno u kompleksnijim aplikacijama [5].

2.4. ECMAScript

ECMAScript (ES) je specifikacija za standardizaciju skriptnih jezika, najčešće za one koji se koriste za klijentsku stranu web aplikacija. Najpoznatija specifikacija je ona za programski jezik JavaScript. ECMAScript standard definira značajke koje JavaScript (ili neki jezik koji se prevodi u JavaScript) mora podržavati da bi zadovoljio tu verziju standarda. Kao što je spomenuto, JavaScript je doživio znatne promjene od svojih ranih dana. Specifikacije su označene brojevima, no kasnije verzije sve češće u nazivu sadrže godinu kad je standard objavljen. U trenutku pisanja ovog seminarskog rada, zadnja objavljena verzija je ECMAScript 8 (ES8), odnosno ECMAScript 2017 (ES2017).

Najznačajnije promjene donio je ES6 standard (poznat i pod nazivom ES2015) koji je u JavaScript uveo razrede, module, lambda funkcije (*arrow functions*), kolekcije, obećanja (engl. *promises*) i mnoge druge funkcionalnosti. Mnoge od tih funkcionalnosti postojale su u drugim programskim jezicima i smatrale su se uobičajenima. Kako je JavaScript narastao od jezika za pisanje manjih skripti na jezik za pisanje složenih dinamičkih aplikacija, razvila se potreba za uvođenjem tih funkcionalnosti.

Neki web preglednici ne podržavaju sve mogućnosti koje su podržane određenim ECMAScript standardom. No, to uglavnom ne predstavlja problem jer se funkcionalnosti s vremenom omoguće, a čak i ako se ne omoguće, postoje alati za prevođenje koda pisanog novim standardom u kod pisan starim standardom. Ti se alati zovu transpilatori (engl. *transpilers*) i najčešće prevode kod u ES5 standard koji je podržan i u starijim web preglednicima kao što su Internet Explorer 11 [6].

2.5. Programski jezik TypeScript

TypeScript je nadogradnja programskog jezika JavaScript. Razvijen je od strane tvrtke Microsoft s ciljem pojednostavljenja razvoja klijentskih web aplikacija. Ideja je bila upotpunjavanje programskog jezika JavaScript s dodacima koje on nema, po uzoru na druge moderne programske jezike. Ključna funkcionalnost TypeScripta u odnosu na JavaScript je podrška za tipove podataka, po čemu je i dobio ime. Također, prati najnovije ECMAScript standarde pa podržava razrede, lambda funkcije i sve nove funkcionalnosti. TypeScript je koristan kao jezik jer olakšava razvoj aplikacija, posebno pri

pronalasku pogrešaka u kodu, budući da JavaScript nudi veliku slobodu i bez tipova je puno teže otkriti uzrok neočekivanog ponašanja aplikacije. Također, mnogim razvojnim programerima lakše je prebaciti se s nekog drugog jezika na TypeScript nego na „običan“ JavaScript jer su navikli na tipove, sučelja i ostale dodatke koje TypeScript unosi u JavaScript, a JavaScript ih (barem za sad) nema. S obzirom da se TypeScript prevodi u JavaScript, pri samom procesu prevođenja gube se sve nadogradnje koje JavaScript nema. No, to ne predstavlja problem jer funkcionalnost ostaje jednaka, a sami dodaci, kao što je rečeno, olakšavaju razvoj aplikacije [7].

3. Angular

3.1. Osnovni podaci

Angular je radni okvir za razvoj klijentskih web aplikacija razvijen od strane tvrtke Google, a verzija 2 objavljena je 2016. godine. Pod nazivom Angular podrazumjevaju se novije verzije ovog radnog okvira (od Angulara 2 nadalje), dok se za prvu verziju koristi naziv AngularJS. AngularJS nije kompatibilan s novim verzijama Angulara jer je Google odlučio iznova napisati cijeli radni okvir. Neka načela koja se koriste u Angularu preuzeta su iz AngularJS-a, no sintaksa je vrlo drukčija i riječ je o dva različita radna okvira. U trenutku pisanja ovog seminarskog rada, zadnja stabilna verzija Angulara je Angular 6. Za razvoj aplikacija koristi se programski jezik TypeScript. Osim TypeScripta, kod je moguće pisati i u jezicima JavaScript i Dart, no manje je uobičajeno [8].

Google koristi Angular za razvoj vlastitih aplikacija, a osim toga koriste ga još i Youtube TV, Udacity, Microsoft Customers, radio.com, Ryanair Rooms, AirAsia, Rockstar Games, Freelancer i NBA.com [9].

3.2. Arhitektura

Arhitektura Angular aplikacije vrlo je razrađena. Podijeljena je na nekoliko logičkih dijelova. Svaki logički element označen je prikladnim dekoratorom[1] i obično je smješten u zasebnu datoteku. Osnovni građevni blok je modul (*NgModules*). Svaka aplikacija ima korijenski modul (engl. *root module*) koji predstavlja kontekst aplikacije i omogućava njeno iscrtavanje (engl. *bootstraping*) te uključuje u aplikaciju druge module koji će se koristiti kao što su komponente i servisi. Komponente i servisi u srži nisu ništa drugo nego razredi označeni svojim pripadnim dekoratorima koji vežu za njih određene funkcionalnosti i služe kao uputa Angularu kako ih treba koristiti [10].

[1] Dekorator (engl. *decorator*) je u Angularu funkcija koja modificira JavaScript razrede. Angular definira različite dekoratore koji opisuju različite tipove razreda.

3.2.1. Komponente

Komponenta (engl. *component*) je zaokružena cjelina i zadužena je za kontrolu svog dijela pogleda (engl. *view*), po uzoru na ideju web komponenti. Aplikacije uglavnom imaju više različitih komponenti koje su hijerarhijski povezane, a njihov broj i organizacija ovise o samoj aplikaciji. Svaka Angular aplikacija ima barem jednu komponentu - korijensku komponentu (engl. *root component*) koja povezuje hijerarhiju ostalih komponenata s DOM-om stranice. Komponente se sastoje se od predloška i pripadnog razreda.

Razred komponente pisan je u jeziku TypeScript i sadrži metapodatke (engl. *metadata*) koji detaljnije opisuju taj razred i povezuju s njim ostale logičke cjeline kao što su predložak, datoteke s dizajnom, uključeni servisi i sl. Osim toga, metapodaci sadrže i naziv oznake s pomoću koje se komponenta može referencirati u drugim predlošcima, tzv. selektor (engl. *selector*). Razred sadrži podatke i logiku vezanu uz kontolu komponente.

Predložak komponente (engl. *template*) pisan je u HTML-u i predstavlja izgled komponente, odnosno opis kako se komponenta treba iscrtati. Može biti odvojen u zasebnu datoteku ili se nalaziti u datoteci u kojoj se nalazi razred (obično ako je predložak manji). Osim uobičajenih HTML oznaka, predložak sadržava i posebnu Angular sintaksu za specifične mogućnosti kao što su direktive (engl. *directives*), povezivanje podataka (engl. *data binding*) ili cjevovodi (engl. *pipes*). Cjevovodi se koriste za transformaciju podataka prije samog prikaza. Predlošci mogu koristiti i selektore drugih komponenata i na taj način iscrtavati njihove predloške unutar sebe [10].

3.2.2. Direktive

Direktive služe za primjenu logike nad predloškom i upravljanje iscrtavanjem podataka. Riječ je također o razredima označenima posebnim dekoratorom za direktive. Vrste direktiva su strukturalna (engl. *structural directive*) i atributska (engl. *attribute directive*). Komponenta je tehnički posebna vrsta direktive, ali se zbog svojih specifičnosti (proširenja predloškom) tretira drukčije. Strukturalne direktive dodaju, uklanjaju ili zamjenjuju elemente predloška, dakle bave se strukturom predloška. Primjeri često korištenih strukturalnih direktiva su ugrađene direktive *NgIf* koja služi za selektivno iscrtavanje HTML elemenata (iscrtavanje ovisi o navedenom uvjetu) i *NgFor* koja služi za iscrtavanje elemenata kolekcije i za uzastopno iscrtavanje istog elementa više puta. Atributske direktive mijenjaju izgled ili ponašanje postojećeg elementa. Primjer ove direktive je npr. ugrađena direktiva *NgStyle* direktiva koja mijenja CSS stil pridružen

elementu [10].

3.2.3. Povezivanje podataka

Angular podržava dvostrano povezivanje podataka (engl. *two-way data binding*). *Data binding* je zapravo mehanizam kojim se koordiniraju podaci u komponenti i podaci iz predloška i kao takav je uobičajen u *desktop* i mobilnim aplikacijama, no u web aplikacijama do nedavno nije bio moguć. Ovaj mehanizam funkcionira tako da se podatak koji se želi prikazati u predlošku veže na konkretnu varijablu, te potom promjena jednog utječe na promjenu drugog. Uglavnom je riječ o tome da promjenom vrijednosti podataka unutar komponente, automatski se mijenjaju podaci prikazani na predlošku, tj. iznova se iscrtavaju jer se dogodila promjena. To je tzv. povezivanje svojstva (engl. *property binding*). Postoji i povezivanje događaja (engl. *event binding*) koji ide u drugom smjeru, tj. neki događaj (engl. *event*) može se povezati s funkcijom koja obrađuje taj događaj (*event handler*) i po potrebi mijenja podatke u komponenti.

Angular podržava oba smjera istovremeno, odnosno *two-way binding*. Dakle, određena varijabla može se vezati na podatak koji se prikazuje u predlošku i promjena vrijednosti tog podatka, bilo u predlošku ili u komponenti, mijenja vrijednost spremljenu u komponenti, odnosno prikaz podatka. Primjer ovog mehanizma može se objasniti npr. upravljanjem *input* elementom. Prikazan tekst je u početku jednak inicijalnoj vrijednosti varijable definiranoj u komponenti. Promijenom varijable unutar komponente, taj se prikaz mijenja (kao kod povezivanja svojstva). Upisom teksta u *input* element predloška, mijenja se vrijednost podatka u komponenti, dakle obrnut smjer [10].

3.2.4. Servisi

Servisi su razredi koji sadrže logiku koja nije vezana ni uz jedan specifični prikaz i koju se dijeli na više mjesta u aplikaciji. Komponenta može uključiti (engl. *inject*) servis unutar sebe koristeći injekciju ovisnosti (engl. *dependency injection*). Na ovaj način komponente mogu delegirati zadatke odgovarajućim servisima koji su specijalizirani za tu zadaću, umjesto da se logika koja nije vezana za komponentu veže na tu komponentu [10].

3.3. Detekcija promjena

Angular koristi posebni mehanizam za praćenje promjena u strukturi DOM-a i njegovo ažuriranje. Struktura svake aplikacije pisane u Angularu interno je reprezentirana s pomoću pogleda. Za svaku instancu komponente (postoje tvornice koje vraćaju instancu određenog tipa komponente), Angular kreira zaseban pogled. Za svaki HTML element i za svaku komponentu generira se čvor pod nazivom definicija elementa (engl. *element definition*). Svaki čvor element može imati za djecu druge čvorove elemente i čvorove definicija teksta (engl. *text definition nodes*), tj. čvorove koji predstavljaju elemente teksta. Angular koristi povezivanje (engl. *binding*) za definiciju ovisnosti (engl. *dependency*) svakog čvora o razrednim svojstvima komponente. Za vrijeme detekcije promjena, svako povezivanje određuje operaciju koju Angular treba izvršiti da bi se čvor ažurirao. [11].

```
1  import {Component} from '@angular/core';
2
3  @Component({
4    selector: 'app-counter',
5    template: '<button (click)="incrementCount()">
6                  Clicks: {{count}}
7               </button>'
8  })
9  export class CounterComponent {
10   count: number = 0;
11
12   incrementCount() {
13     this.count++;
14   }
15 }
```

Isječak koda 3.1: Primjer Angular komponente koja na zaslon iscrtava gumb koji prikazuje koliko se puta kliknulo na njega.

4. React

4.1. Osnovni podaci

React (ReactJS ili React.js) je biblioteka za razvoj klijentskih web aplikacija u programskom jeziku JavaScript. Razvijen je od strane tvrtke Facebook, a prva verzija izašla je 2013. godine. U trenutku pisanja ovog seminarskog rada, zadnja stabilna verzija Reacta je React 16. React po definiciji dakle nije radni okvir već biblioteka. Da bi se u potpunosti mogao klasificirati kao radni okvir, potreban mu je mehanizam upravljanja stanjem (engl. *state management*), odnosno, promatrajući MVC obrazac, React obuhvaća samo dio MVC obrasca koji se odnosi na pogled. Za potpunu implementaciju MVC obrasca, obično se koristi neki drugi radni okvir kao što su Redux, Flux, MobX i dr. Osim za razvoj klijentskih web aplikacija (konkretno aplikacija jedne stranice), može se koristiti i za razvoj nativnih mobilnih aplikacija (Android i iOS) u svojoj inačici React Native [12].

Neke od poznatih aplikacija koje koriste React su Facebook, Instagram, Airbnb, Uber, Netflix, Twitter, Reddit, Wix, Paypal, Imgur, i Tumblr [13].

4.2. Arhitektura

React aplikacija sastoji se od komponenti (engl. *components*). Komponente su međusobno hijerarhijski organizirane. Postoje funkcijske (engl. *functional components*) i razredne komponente (engl. *class components*). Funkcijska komponenta, kao što i samo ime kaže, predstavljena je JavaScript funkcijom. Ta funkcija vraća React element koji se treba iscrtati, a prima parametar *props* (skraćeno od *properties* - svojstva). Razredna komponenta zapisana je u obliku ES6 razreda. Svaka razredna komponenta sadrži metodu *render* koja vraća React element koji se treba iscrtati. Funkcijska i razredna komponenta su iz perspektive Reacta jako slične, odnosno svaka funkcijska komponenta može se zapisati u obliku razredne.

Funkcijska komponenta ima ograničene funkcionalnosti u odnosu na razrednu komponentu. Ključna razlika između funkcijske i razredne komponente je to što razredna komponenta može imati stanje. Stanje je predstavljeno *state* objektom koji sadrži svojstva koja se mogu mijenjati na odgovarajući način i time utjecati na ponašanje komponente, prikaz podataka i sl. Može ga vidjeti i mijenjati samo komponenta koja ga posjeduje i zbog toga se često kaže da je stanje lokalno, odnosno enkapsulirano. Stanje komponente usko je povezano sa životnim ciklusom komponente o kojem će biti više riječi kasnije. S obzirom na to sadrži li komponenta stanje, razlikujemo komponente sa stanjem (engl. *stateful components*) i komponente bez stanja (engl. *stateless components*) [14].

4.2.1. JSX

JSX je posebna sintaksa za zapis kako se aplikacija treba iscrtati. Nalikuje HTML-u zbog toga što sadrži oznake, no zapravo nije riječ o HTML-u već o JavaScriptu. Kada se detaljnije pogleda, može se primjetiti da su, primjerice, imena atributa oznaka drukčija nego u HTML-u, tj. zapisana su u *camel-case* formatu. JSX je zapravo ljepši način za zapis JavaScript funkcija koje su u Reactu zadužene za kreiranje React elemenata. React elementi nisu ništa više nego objekti koji opisuju što se treba prikazati na zaslonu, dakle opisuju krajnju strukturu DOM-a. Nakon prevođenja, JSX postaje običan JavaScript. To znači da ga je moguće miješati s kodom pisanim u JavaScriptu, što se čini neobično, no sasvim je normalno za React. Također je moguće pisati JavaScript kod unutar JSX-a, bitno je samo izdvojiti ga korištenjem vitičastih zagrada. JSX je jedna od najvećih specifičnosti Reacta. React na ovaj način elegantno miješa kod zadužen za kontrolu DOM-a i kod zadužen za prikaz podataka i nudi veliku fleksibilnost. Naposlijetku, u Reactu je sve JavaScript. Ako se nekome i ne sviđa ova sintaksa, nije ju obvezan koristiti već uvijek može pisati ekvivalentne JavaScript funkcije. Također, korištenje JSX-a u odnosu na HTML nudi neke prednosti, kao što su prevencija napada ubacivanja koda (engl. *injection attacks*) budući da se React prije iscrtavanja DOM-a pobrine za to [14].

4.2.2. Povezivanje podataka

React podržava povezivanje svojstva (engl. *property binding*) i povezivanje događaja (engl. *event binding*). Dakle, za razliku od Angulara, ne podržava dvosmjerno povezivanje (engl. *two-way binding*) [14].

4.3. Detekcija promjena

React koristi virtualni DOM za praćenje promjena u stanju komponenti i ažuriranje DOM-a.

4.3.1. Virtualni DOM

Virtualni DOM (engl. *Virtual DOM*) je programski koncept u kojem se u memoriji čuva „virtualna“ reprezentacija DOM-a i sinkronizira se s pravim DOM-om. U Reactu se za sinkronizaciju virtualnog DOM-a sa stvarnim DOM-om koristi biblioteka React-DOM. Korištenjem ovog mehanizma, React omogućuje deklarativan pristup korištenju svog sučelja. Dakle, programer postavi stanje u kojem želi da se komponenta nalazi, a React se sam pobrine za prikaz podataka, odnosno da stvarni DOM odgovara tom stanju. Na ovaj način se apstrahiraju mehanizmi ažuriranja DOM-a pa se programer ne mora brinuti o tome.

Tradicionalno ažuriranje DOM-a je vrlo spor proces. Virtualni DOM u Reactu funkcionira tako da se pri promjeni stanja unutar neke komponente ta komponenta označi kao „prljava“ (engl. *dirty*) i treba se ponovno iscrtati. Dakle, pri svakoj promjeni stanja, React kreira novi virtualni DOM. Kreiranje novog stabla za pohranu virtualnog DOM-a je brz postupak pa ne narušava performanse. U svakom trenutku postoje dva virtualna DOM-a, jedno sa starim vrijednostima i jedno sa vrijednostima nakon promjene stanja. React koristi efikasan algoritam za usporedbu ta dva virtualna DOM-a kako bi pronašao najmanji potreban broj koraka za ažuriranje stvarnog DOM-a. Ovaj je pristup brži od tradicionalnog ažuriranja DOM-a i jedan je od razloga dobrih performansi Reacta [14] [15].

```
1  export class Counter extends React.Component {
2    constructor(props) {
3      super(props);
4      this.state = { count: 0 };
5    }
6
7    incrementCount = () => {
8      this.setState({ count: this.state.count + 1 });
9    };
10
11   render() {
12     return (
13       <button onClick={ this.incrementCount }>
```

```
14          Clicks: { this.state.count }
15        </button>
16      );
17    }
18  }
```

Isječak koda 4.1: Primjer React komponente koja na zaslon iscrtava gumb koji prikazuje koliko se puta kliknulo na njega.

5. Vue

5.1. Osnovni podaci

Vue (VueJS ili Vue.js) je trenutno najmanje poznat JavaScript radni okvir među navedenima, no njegova popularnost svakim danom raste. Razvio ga je Evan You, bivši zaposlenik tvrtke Google, nakon mnogo iskustva rada s AngularJS-om. Prvi put je objavljen 2014. godine. Cilj mu je bio iskoristiti ono najbolje što AngularJS nudi u jednom manjem radnom okviru, bez nepotrebnih dodataka. Iako ima sličnosti i s Reactom, za razliku od njega, Vue je zbog svog ugrađenog sustava upravljanja stanjem (engl. *state management*) pod nazivom Vuex pravi radni okvir [16] [17]. Danas Vue razvija internacionalni tim koji čini i njegov originalni autor [18]. U trenutku pisanja ovog seminarskog rada, zadnja stabilna verzija VueJS-a je Vue 2.

Primjeri tvrtki i aplikacija koje koriste Vue su Adobe, Alibaba, Xiaomi, Grammarly i GitLab [19].

5.2. Arhitektura

Svaka Vue aplikacija započinje kreiranjem Vue instance (engl. *root Vue instance*) korištenjem Vue funkcije. Vue instanci se pri kreiranju predaje objekt s opcijama (engl. *options object*). Osim objekta s opcijama, instanca može sadržavati i objekt s podacima (engl. *data object*) čija se svojstva pri kreiranju instance dodaju u sustav reaktivnosti (engl. *reactivity system*). Dakle, kad god se vrijednosti tih svojstava promijene, pogled aplikacije će se izmijeniti u skladu s promjenjenim podacima. U sustav reaktivnosti dodaju se samo svojstva koja su postojala pri kreiranju instance, tj. naknadno dodana svojstva neće se promatrati.

VueJS aplikacija izgrađena je od komponenti (engl. *component*), a komponente su zapravo posebne vrste Vue instanci (imenovane Vue instance koje potom mogu više puta koristiti). U VueJS-u, slično kao u Angularu, komponente sadrže predloške (engl.

template) [20].

Predlošci su pisani u HTML-u u kombinaciji s dodatnom Vue sintaksom (slična ideja kao u Angularu). Osim toga, Vue također podržava i sintaksu za pisanje funkcija za iscrtavanje i JSX poput Reacta, iako razvojni programeri VueJS-a smatraju da je jednostavnije koristiti uobičajene predloške. Mnogim je programerima prirodnije čitati i pisati HTML kod (nadopunjen dodatnom sintaksom), a također je puno lakše i dizajnerima i ostalima koji imaju manje programerskog iskustva. Uz to, HTML kod starijih aplikacija lagano je prebaciti u Vue predloške [21].

Vue sintaksa za predloške uključuje direktive (engl. *directive*), povezivanje podataka (engl. *data binding*) i filtere (engl. *filters*). Direktive su slične onima u Angularu. Kao i u Angularu, moguće je kreirati vlastite direktive. Filteri su analogni cjevovodima u Angularu [20].

5.2.1. Povezivanje podataka

Vue, kao i Angular, podržava dvostrano povezivanje podataka (engl. *two-way data binding*). Dakle, povezivanje podataka može ići iz smjera objekta podataka prema predlošku (*property binding*), iz smjera predloška reakcijom na događaj (*event binding*), ili obostrano korištenjem dvostranog povezivanja [20] [21].

5.3. Detekcija promjena

Vue, poput Reacta, koristi virtualni DOM, unatoč tome što koristi HTML predloške. To je moguće zbog toga što je Vue razvio mehanizam automatskog prevođenja predloška u virtualni DOM. Iako je ideja slična, za razliku od Reactovog, VueJS-ov virtualni DOM automatski prati ovisnosti komponente (engl. *dependencies*) za vrijeme njenog iscrtavanja, tako da sustav točno zna koje komponente treba ponovno iscrtati pri promjeni stanja [20].

```
1 Vue.component('button-counter', {
2   data: function () {
3     return {
4       count: 0
5     }
6   },
7   template: '<button v-on:click="count++">
8                 Clicks: {{ count }}
9             </button>'
```

```
10 })
```

Isječak koda 5.1: Primjer Vue komponente koja na zaslon iscrtava gumb koji prikazuje koliko se puta kliknulo na njega.

6. Usporedba radnih okvira

Sva tri radna okvira otvorenog su koda (engl. *open-source*), zaštićeni su MIT licencom i njihovi su kodovi dostupni na stranicama GitHuba. Na taj način zajednice programera mogu dati svoj doprinos razvoju tih radnih okvira.

React i Angular oko sebe imaju puno veći ekosustav nego što to ima Vue. Mogući uzrok tome je to što iza Reacta i Angulara stoje velike tvrtke koje šire popularnost vlastitih radnih okvira. Posebno velik ekosustav ima React budući da nije potpun radni okvir pa velik broj programera piše svoje biblioteke za podršku funkcionalnosti koje React sam po sebi ne podržava.

Općenito, dobro je kada postoji velik ekosustav i velika zajednica programera koji koriste neku tehnologiju jer to znači da će uvijek postojati netko koga se može pitati za pomoć. Dakle, podrška zajednice igra veliku ulogu u izboru radnog okvira.

6.1. Složenost arhitekture

Angular ima najsloženiju arhitekturu. Arhitektura Angulara je vrlo razrađena, no zbog toga je Angular puno veći radni okvir i ujedno teži za naučiti. Veličina radnog okvira igra ulogu i pri dohvaćanju koda sa poslužitelja. Angular tim je koristeći razne metode uspio značajno smanjiti veličinu produkcijskog koda, no i dalje je poprilično velik.

Vue i React su puno manji i jednostavniji radni okviri, što ih čini lakšima za učenje i fleksibilnijima jer nemaju toliko strogo zadana pravila kao Angular.

No, je li strogo definirana arhitektura dobra ili ne je donekle i stvar ukusa. Neki programeri više vole stroga pravila i jednoznačno dobro rješenje, dok drugi više cijene fleksibilnost u programiranju.

6.2. Broj korisnika i njihovo zadovoljstvo

Usporedit ćemo radne okvire po broju programera koji ih koriste i njihovom zadovoljstvu u korištenju istih te želji za učenjem tih radnih okvira od strane programera koji

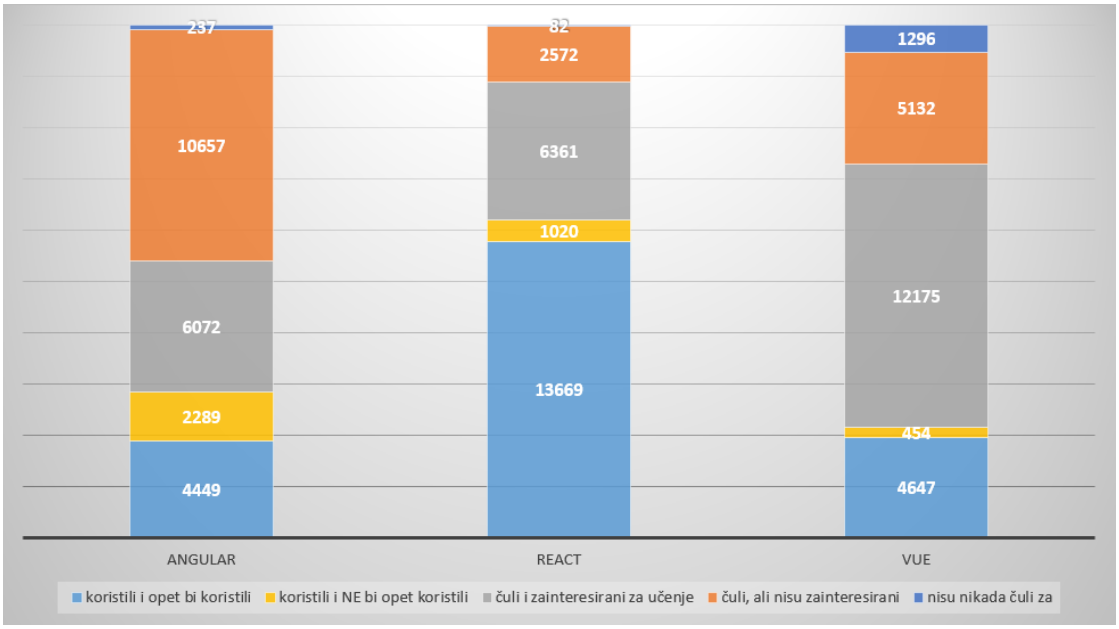
ih nisu koristili. Na dijagramu na slici 6.1 prikazani su prikupljeni podaci.

Prema rezultatima ankete *The State of JavaScript 2017* provedenoj 2017. godine [22] među programerima diljem svijeta, najpopularniji radni okvir (zapravo biblioteka) je React. Velik broj programera je već koristio React i svidio im se, a oni koji nisu su se većinom izjasnili da bi ga voljeli naučiti.

Sljedeći po broju programera koji su isprobali taj radni okvir je Angular 2. No, veći postotak njih se izrazio da ga više ne bi koristili, iako je i dalje veći postotak programera koji ga žele koristiti. Među programerima koji nisu koristili Angular, no čuli su za njega, više je programera koji ne žele naučiti Angular. Razlog ovoj niskoj motivaciji za učenjem Angulara su možda postojeće jednostavnije alternative kao što su React i Vue. Moguće je i da se dio tih programera razočarao u AngularJS i zbog toga ne žele dati Angularu priliku. Za AngularJS je broj programera koji su ga koristili i ne žele ga više koristiti puno veći od broja onih koji ga i dalje žele koristiti.

Vue je po rezultatima ankete najmanje poznat radni okvir od ova tri radna okvira, odnosno najmanje je programera čulo za njega. No, među onima koji su čuli za njega, većina je zainteresirana za učiti Vue. Od programera koji su isprobali Vue, velika većina ga želi i dalje koristiti.

Dakle, od ova tri radna okvira, React se trenutno najviše koristi, popularnost Angulara malo opada, dok VueJS-ova popularnost raste.



Slika 6.1: Rezultati ankete *The State of JavaScript 2017*

6.3. Broj ocjena na GitHub repozitoriju

Stranica *Rising Stars* usporedila je JavaScript radne okvire prema broju ocjena, tj. zvijezdi koje su dobili u 2017. godini na svojim GitHub repozitorijima. Zvijezdu može dodijeliti osoba koja ima kreiran GitHub profil, i to samo jednom za isti repozitorij, a dodjeluje ju ako joj se sviđa repozitorij.

Prva tri mjesta zauzeli su upravo radni okviri analizirani u ovom seminarskom radu. Na prvom je mjestu Vue s 40 tisuća prikupljenih zvijezdi kroz 2017. godinu. Drugo mjesto zauzeo je React s 27.8 tisuća dobivenih zvijezdi, dok je na trećem mjestu Angular s 12.2 tisuće zvijezdi [23].

Ovi rezultati pokazuju kako je popularnost VueJS-a doista narasla kroz proteklu godinu. No, i druga dva radna okvira su i dalje vrlo popularan izbor.

7. Zaključak

Svaki od navedena tri radna okvira dobar je na svoj način i svaki od njih ima svoje prednosti i mane.

Angular je vrlo kompleksan i teži za naučiti od preostala dva suparnika. Zbog toga ga programeri često izbjegavaju jer ne žele ulagati previše vremena u učenje Angulara kad im je lakše naučiti neki lakši radni okvir. Također, za jednostavnije aplikacije je ponekad bolje koristiti radni okvir koji je manji jer se manje podataka treba učitati što čini aplikaciju bržom. Mnogi još uvijek miješaju AngularJS i Angular i zbog lošeg iskustva s AngularJS-om ili loših mišljenja koja su čuli o njemu ne žele isprobati Angular koji je posve drukčiji od AngularJS-a. No, Angular je dobar za velike aplikacije s puno logike jer je vrlo organiziran i sadrži velik broj ugrađenih funkcionalnosti.

React je jednostavan i specijaliziran za prikaz podataka, no s obzirom na to da nije pravi radni okvir nego biblioteka, nudi puno manje funkcionalnosti. Za imalo kompleksnija rješenja, potrebno je uključivati mnoge druge biblioteke i to može otežati razvoj aplikacija. No, s druge strane, za većinu manje kompliciranih aplikacija predstavlja dobar izbor. React zauzima malo prostora i zbog toga se aplikacije pisane u Reactu brzo učitavaju. To dokazuje i njegova velika popularnost - React je popularniji od svoja dva suparnika.

Vue je nova zanimljiva stvar u JavaScript svijetu koja tek od nedavno dobiva sve više pažnje. Za razliku od Angulara i Reacta, iza VueJS-a ne stoji niti jedna velika tvrtka koja bi ga gurala naprijed. Vue oko sebe još uvijek ima malu zajednicu programera što može predstavljati problem jer je manji broj ljudi koji stoje na raspolaganju za dati savjet ili pomoći oko problema. Angular i React imaju svoje velike zajednice i mnogo objava na raznim blogovima i stranicama za pomoć oko programiranja. No, bez obzira na to, Vue danas predstavlja veliku konkurenciju Angularu i Reactu. Razlog tome je što je Vue naučio mnogo od drugih radnih okvira i implementirao je njihova dobra svojstva. Rezultat je radni okvir koji je potpun, no nije prevelik, a nudi sve što je potrebno većini današnjih web aplikacija. Uz to je jednostavan i fleksibilan. Vrijeme će pokazati hoće li popularnost VueJS-a rasti.

8. Literatura

[1] Wikipedia – The Free Encyclopedia: JavaScript. https://en.wikipedia.org/wiki/JavaScript. Datum pristupa: 17.05.2018.

[2] Wikipedia – The Free Encyclopedia: Single-page application. https://en.wikipedia.org/wiki/Single-page_application. Datum pristupa: 17.05.2018.

[3] Mozilla Developer Network – Web Components. https://developer.mozilla.org/en-US/docs/Web/Web_Components. Datum pristupa: 17.05.2018.

[4] Web Components. https://www.webcomponents.org/. Datum pristupa: 17.05.2018.

[5] Learn How to Program – "Data Down, Actions Up". https://www.learnhowtoprogram.com/javascript/angular/data-down-actions-up. Datum pristupa: 17.05.2018.

[6] Wikipedia – The Free Encyclopedia: ECMAScript. https://en.wikipedia.org/wiki/ECMAScript. Datum pristupa: 17.05.2018.

[7] TypeScript. https://www.typescriptlang.org/. Datum pristupa: 17.05.2018.

[8] Wikipedia – The Free Encyclopedia: Angular (application platform). https://en.wikipedia.org/wiki/Angular_(application_platform), . Datum pristupa: 17.05.2018.

[9] Lalith Polepeddi. Made with Angular. https://www.madewithangular.com/categories/angular. Datum pristupa: 17.05.2018.

[10] Angular – Architecture overview. https://angular.io/guide/architecture, . Datum pristupa: 17.05.2018.

[11] Maxim Koretskyi. Angular in Depth – The mechanics of DOM updates in Angular. https://blog.angularindepth.com/the-mechanics-of-dom-updates-in-angular-3b2970d5c03d. Datum pristupa: 17.05.2018.

[12] Wikipedia – The Free Encyclopedia: React (JavaScript library). https://en.wikipedia.org/wiki/React_(JavaScript_library), . Datum pristupa: 17.05.2018.

[13] Sites Using React – facebook/react Wiki. https://github.com/facebook/react/wiki/Sites-Using-React, . Datum pristupa: 17.05.2018.

[14] React Docs. https://reactjs.org/docs/, . Datum pristupa: 17.05.2018.

[15] Rupesh Mishra. Hacker Noon – Virtual DOM in ReactJS. https://hackernoon.com/virtual-dom-in-reactjs-43a3fdb1d130. Datum pristupa: 17.05.2018.

[16] Wikipedia – The Free Encyclopedia: Vue.js. https://en.wikipedia.org/wiki/Vue.js, . Datum pristupa: 17.05.2018.

[17] O. Filipova. *Learning Vue.js 2: Learn how to build amazing and complex reactive web applications easily with Vue.js*. Birmingham. Packt Publishing, 2016.

[18] Meet the Team – Vue.js. https://vuejs.org/v2/guide/team.html, . Datum pristupa: 17.05.2018.

[19] Michał Sajnóg. 13 Top Companies That Have Trusted Vue.js – Examples of Applications. https://www.netguru.co/blog/13-top-companies-that-have-trusted-vue.js-examples-of-applications. Datum pristupa: 17.05.2018.

[20] Introduction – Vue.js. https://vuejs.org/v2/guide/index.html, . Datum pristupa: 17.05.2018.

[21] Comparison with Other Frameworks – Vue.js. https://vuejs.org/v2/guide/comparison.html, . Datum pristupa: 17.05.2018.

[22] The State of JavaScript 2017 – Front-end Frameworks – Results. https://stateofjs.com/2017/front-end/results. Datum pristupa: 17.05.2018.

[23] 2017 JavaScript Rising Stars. https://risingstars.js.org/2017/en/#section-framework. Datum pristupa: 17.05.2018.

9. Sažetak

Cilj seminarskog rada je dati pregled i napraviti usporedbu trenutno najkorištenijih modernih web frontend radnih okvira. Na početku se iznose zajednički koncepti radnih okvira Angular, React i Vue. Zatim se obrađuju posebnosti svakog od navedenih radnih okvira, odnosno napravi se pregled spomenutih radnih okvira. Potom se uspoređuju njihove ključne sličnosti i razlike.