Razvoj klijentskog dijela web aplikacije

Šubić, Lorena

SVEUČILIŠTE U ZAGREBU
FILOZOFSKI FAKULTET
ODSJEK ZA INFORMACIJSKE I KOMUNIKACIJSKE ZNANOSTI
Ak. god. 2021./ 2022.

Lorena Šubić

Razvoj klijentskog dijela web aplikacije

Završni rad

Mentor: dr.sc. Vedran Juričić, doc.

Zagreb, lipanj 2022.

Izjava o akademskoj čestitosti

Izjavljujem da je ovaj rad rezultat mog vlastitog rada koji se temelji na istraživanjima te objavljenoj i citiranoj literaturi. Izjavljujem da nijedan dio rada nije napisan na nedozvoljen način, odnosno da je prepisan iz necitiranog rada, te da nijedan dio rada ne krši bilo čija autorska prava. Također izjavljujem da nijedan dio rada nije korišten za bilo koji drugi rad u bilo kojoj drugoj visokoškolskoj, znanstvenoj ili obrazovnoj ustanovi.

Htjela bih se zahvaliti svojem Dragom mentoru bez kojega ne bih uspjela napisati ovaj završni rad, te T.K. i M.L. što su uvijek bile uz mene.

Sadržaj

Popis kodova

Popis tablica

Popis slika

Popis grafova

1. Uvod

U radu će se prikazati komponente aplikacije rađene u klijent-server arhitekturi, a poseban naglasak će se staviti na razvoj klijentskog dijela web aplikacije. Klijent-server tehnologija je računalni model koji odvaja funkcionalnost klijenta i poslužitelja, koji su uglavnom međusobno povezani pomoću računalne mreže. Pritom, svaka instanca klijenta može poslati zahtjeve za podacima jednom od poslužitelja na mreži i pričekati odgovor, a za uzvrat, neki od dostupnih poslužitelja može prihvatiti navedene zahtjeve, obraditi ih i vratiti rezultat klijentu.

Web aplikacija je interaktivni računalni program izgrađen s web tehnologijama poput (HTML-a, CSS-a i JavaScripta), koji podatke pohranjuje u bazu podataka, datoteke i sl. i manipulira njima, a koristi ga jedan ili više korisnika za obavljanje zadataka. Web aplikacije se sastoje od backenda i frontenda. Backend je serverski dio aplikacije koji realizira traženu uslugu. Backend razvojni inženjeri kreiraju aplikacije koje su dostupne na strani poslužitelja, rade sa softverskim stogovima koji, između ostalog, uključuju web poslužitelje, okvire, operacijske sustave, programske jezike, aplikacijska programska sučelja (API, *Application Programming Interface*). Softverski stogovi uključuju .NET, MEAN i LAMP, te svaki uključuje određeni programski jezik, npr. JavaScript, C#, Go, Python i PHP. Frontend razvoj je stvaranje web mjesta i web aplikacija koje se prikazuju na strani klijenta.

Frontend razvojni inženjer odgovoran je za realizaciju dizajna korisničkog sučelja proizvoda, odnosno za pretvaranje datoteka s dizajnom u odgovarajući programski kod. Na klijenskoj strani se za razvoj koriste tehnologije poput HTML-a, CSS-a i JavaScripta. Cilj dizajna web mjesta je osigurati da korisnici kada otvore web mjesto vide informacije u formatu koji je jednostavan za čitanje i relevantan. Iako se klijentski dio temelji na tehnologijama poput HTML-a, CSS-a i JavaScripta, razvojni inženjeri vrlo često koriste okvire koji pružaju moderan pristup dizajnu i funkcionalnosti, te olakšavaju prikaz podataka dohvaćenih sa servera.

U radu će se analizirati popularni okviri, njihove prednosti, nedostaci i značajke, kao i mogućnosti njihove integracije u postojeći sustav. Također će se analizirati sličnosti i razlike u korištenju i sintaksi najpoznatijih okvira, poput AngularJS koji je samostalni okvir s ugrađenim alatima i knjižnicama koje ne ometaju veličinu ili brzinu aplikacije, Vue.js koji je okvir temeljen na JavaScriptu i koji pruža napredne alate za razvoj modernih aplikacija, te React.js je biblioteka za razvoj korisničkog sučelja temeljena na JS, te ga pokreće Facebook i zajednica razvojnih inženjera otvorenog koda.

2. Tehnologije

Web je zasnovan na klijent – server modelu komunikacije. Korištenje web stranica preko web preglednika isto je što i korištenje sadržaja kojeg bilo koji korisnik vidi kao tekst, fotografiju ili video. Prilikom korištenja web stranica preko web preglednika. Sve što vidimo na webu je kombinacija HTML-a, CSS-a i JavaScript jezika. Navedena su tri glavna jezika korištena za predstavljanje i prezentiranje web platforme na najbolji mogući način (Hill i Brannen, 2011). HTML je odgovoran za stvaranje osnovne strukture web platforme. Preko CSS-a se uvode boje, veličine fonta, pozadinu, margine, pozicije elemenata itd. JavaScript je programski jezik poznat po dinamičkim mogućnostima, omogućava interaktivnost, poput pomicanja, klikanja itd.

2.1. HTML

Ideja iz koje se razvio World Wide Web nastala je 1989. godine kao posljedica potrebe za razmjenom informacija. Radom na mreži su se u to vrijeme najviše koristili znanstvenici. Tim Berners – Lee predložio je kreiranje hipertekstualnog sustava koji će olakšati razmjenu informacija među stručnjacima i pritom je nastao HyperText Markup Language (HTML) (Piligram, 2010). HTML 1.0 objavljen je 1993. godine s namjerom razmjenjivanja informacija koje mogu biti razumljive i dostupne svima pomoću web preglednika. 1995. je izašao HTML 2.0 koji sadrži sve značajke HTML-a 1.0 zajedno s nekoliko dodatnih značajki, koje su ostale kao standardni označni jezik za dizajn i kreiranje web stranica. U HTML-u 3.0 je Dave Raggett predstavio novi rad ili nacrt o HTML-u, a 2012. je izašao HTML 4.01 koji se vrlo često koristi i koji je bio uspješna verzija HTML-a prije verzije 5.0. Ova verzija je objavljena 2014. godine i upotrebljava se širom svijeta. U jednom razdoblju su nove verzije izlazile toliko često da je bilo nemoguće pročitati knjigu i dokumentaciju postojeće verzije (Delamater i Ruvalcaba, 2020).

HTML je sustav oblikovanja za prikaz materijala dohvaćenog putem Interneta. HTML oznake za označavanje određuju elemente dokumenta, kao što su naslovi, odlomci i tablice, koje će se prikazati pomoću računalnog programa, odnosno web preglednika. Preglednik interpretira oznake prikazujući naslove, odlomke i tablice u izgledu koji je prilagođen veličini zaslona i fontovima koji su mu dostupni (Harris, 2014). Za pisanje i pregled HTML dokumenta nije potrebna veza na Internet. Služi isključivo za uređivanje izgleda korisnikove web stranice, te se njime ne mogu izvršavati nikakve matematičke operacije. HTML se može pisati u

uređivaču teksta poput Notepada, Notepad++-a, a danas se za pregled stranice koriste web preglednici poput Chromea, Firefoxa, itd.

HTML oznaka definira početak i završetak nekog elementa, dok atribut opisuje njegove karakteristike (Boehrm i Ruvalcaba, 2022). Neki elementi su prazni, odnosno nemaju sadržaj, ali se sve što je potrebno za prikaz može definirati pomoću atributa. HTML nije osjetljiv na mala i velika slova, ali preporuka je koristiti mala slova za nazive oznaka.

HTML datoteka sadrži tri osnovne oznake: html, head i body. Svaki od njih ima različitu funkciju: html oznaka definira početak i završetak koda, head definira zaglavlje stranice, dok body definira glavni dio, odnosno tijelo stranice. Kao što je prikazano u Kodu 1. osnovna struktura započinje oznakom deklaracije vrste dokumenta <!DOCTYPE>, nakon deklaracije slijedi <html> i završava </html>, a unutar navedenog elementa nalaze se dva glavna dijela: <head> </head> i <body> </body>.

```
<!DOCTYPE html>
<html>
  <head>

  </head>
  <body>

  </body>
</html>
```

Kod 1. *Prikaz HTML dokumenta 1.0*

HTML elementi su kombinacija HTML oznaka i sadržaja. HTML element je između početne i završne oznake (McGrath, 2020). Primjeri elemenata u zaglavlju dokumenta su prikazani u Tablica 1.

Tablica 1. *HTML elementi zaglavlja*

Element	**Objašnjenje**
<title>	Definira naslov dokumenta
<meta>	Definira meta informacije
<style>	Deklarira stilove za ostale elemente
<base>	Definira osnovnu adresu za ostale linkove

Primjeri elemenata u tijelu dokumenta su prikazani u Tablica 2.

Tablica 2. *HTML elementi tijela*

Element	**Objašnjenje**
<h1> do <h6>	Naslovi u dokumentu
<p>	Paragraf teksta
 	Prijelom teksta
<b>	Podebljani tekst
<img>	Slika
<table>	Tablica
<a>	Veza na drugi dokument ili područje unutar istog
<form>	Forma za unos podataka
<select>	Padajući izbornik
<button>	Gumb
<textarea>	Područje za unos teksta više linija

Slika 1 prikazuje HTML kod koji uključuje naslov dokumenta, glavni naslov stranice, podnaslov stranice, paragraf teksta i sliku. S desne strane je izgled navedene stranice u pregledniku.

```html
<!DOCTYPE HTML>
<html lang="hr">
 <head>
    <title>Naslov dokumenta</title>
  </head>

  <body>
    <h1>Glavni naslov</h1>
    <h2>Podnaslov</h2>

    <p>Ovo je paragraf nekog teksta</p>

    <img src="vir.jpg" alt="Vir" width="400" height="500">

  </body>
</html>
```

Slika 1. *Prikaz HTML dokumenta 1.0*

2.2. CSS

CSS (Cascading Style Sheets) je jezik za opis prezentacije web stranica, uključujući boje, struktura i fontove (Harris, 2014). Omogućuje prilagodbu prezentacije različitim vrstama uređaja, kao što su veliki zasloni, mali zasloni ili pisači. CSS je neovisan o HTML-u i može se koristiti s bilo kojim označnim jezikom koji se temelji na XML-u (Meyer, 2018). Odvajanje HTML-a od CSS-a, odnosno sadržaja od prezentacije, olakšava održavanje, dijeljenje stilova među stranicama i prilagodbu stranica različitim okruženjima.

Web preglednici vrlo dugo nisu dosljedno implementirali CSS specifikaciju pa se autori nisu mogli pouzdati da će stranice izgledati približno jednako u svim web preglednicima. Razvijeni su brojni metode čija je namjena bila da isprave neočekivana ponašanja u nekim preglednicima, ali je današnja situacija puno bolja, iako se i danas savjetuje provjera izgleda stranice u što više različitih preglednika (Meyer, 2018). Kod CSS-a je interesantno što definira pravila u stilskom obrascu koji određuje kako se neki sadržaj opisan određenim HTML kodom povezuje sa stilskim pravilima i HTML kodom (Goodman, 2002). Dizajn izgleda stranice u HTML-u, ograničen je na tablice, fontove i nekoliko stilova poput Bold i Italic. Sa stilskim obrascima uz pomoć CSS-a može se kontrolirati svaki aspekt prikaza na stranici (razmak između linija, znakova, margina stranica, pozicije slike itd.).

Korištenjem CSS obrasca moguće je kontrolirati bilo koji dio segmenta web stranice, kao definiranje pozadine (boja pozadine, slika na pozadini), uređivanje listi, uređivanje

margina, kontrola pozicioniranja elemenata, opcije klasifikacija, odnosno kontrole načina ponašanja elemenata na stranici u odnosu na druge elemente, kontrola količine praznog prostora oko bilo kojeg blok elementa na stranici (engl. *padding*), uređivanje tablice i teksta (McGrath, 2020). Sintaksa CSS-a je vrlo jednostavna i sastoji se od dva elementa: selektora i deklaracijskog bloka. Selektor određuje element na koji se stilsko pravilo odnosi, a deklaracijski blok određuje kako izgleda sadržaj opisan CSS-om. Stilski obrasci nadjačavaju unutarnja pravila prikaza u web pregledniku. Za određivanje stilskog pravila koriste se interpunkcijski i posebni znakovi, tako da sintaksa za stilska pravila uvijek slijedi navedeni uzorak:

selektor {deklaracija;}

Deklaracija se sastoji od svojstva (engl. *properties*), što se odnosi na način prikaza grafike i teksta, i na vrijednosti (engl. *values*), koji predstavljaju podatke koji definiraju izgled slike i teksta na web stranici. U deklaraciji svojstva se od vrijednosti odvaja dvotočkom, a točkom-zarez se završava svaka deklaracija (Robbins, 2018):

selector {property: value;}

Kao što je prikazano u Kodu 2., stilskih pravila možemo napraviti nekoliko za jedan selektor, svaki s jednom deklaracijom, no toliko velike kolekcije stilskih pravila su teške za upotrebu. CSS nam dozvoljava da u jedno stilsko pravilo kombiniramo nekoliko deklaracija koje utječe na karakteristike prikaza pojedinom selektora, kao što je prikazano u Kodu 3.

```
1  h1 {color: teal;}
2  h1 {font-family: Arial;}
3  h1 {font-size: 36px}
```

Kod 2. *Prikaz CSS dokumenta 1.0*

```
1 ▼ h1{
2      color: teal;
3      font-family: Arial;
4      font-size: 36px;
5  }
```

Kod 3. *Prikaz CSS dokumenta 2.0*

Važno je za naglasiti kako je ugnježđivanje elemenata jedna od važnijih stvari u HTML-u. Naime, CSS specifikacije taj dio prepoznaju da ugniježđeni elementi, jedan unutar drugoga mogu osigurati povezivanje stilova s osnovnim elementom i upravo tako da se nađe put do elementa djeteta. U HTML svijetu se to popularno naziva nasljeđivanje (Goodman, 2002). Dakako, treba naglasiti da kada se određenom elementu dodaje njegov stil, isti stil primjenjuje se na sve elemente ugniježđene unutar nekog drugog elementa. Uz to, stilska pravila žele postaviti za cijeli dokument, mora ih se zadati u *body* elementu kao što je prikazano u Kodu 4 (Meyer, 2018).

```
<!DOCTYPE html>
<html lang="hr">
 <head>
    <title>Naslov dokumenta</title>
 </head>

  <style>

  body {
    background: teal;
    color: yellow;
    font-size: 18px;
    font-family: Garamond;
    margin-left: 72px;
    }
  </style>
 <body>
   <h1>Glavni naslov</h1>
   <h2>Podnaslov</h2>

   <p>Ovo je paragraf nekog teksta</p>

   <img src="vir.jpg" alt="Vir" width="200" height="250">

 </body>
</html>
```

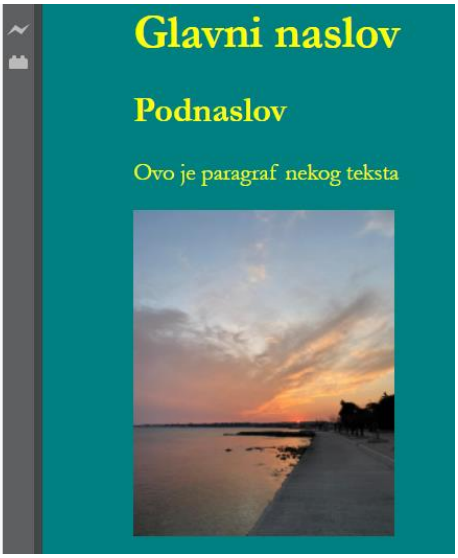
Kod 4. *Prikaz CSS dokumenta 3.0*

Vanjski stilski obrazac ima definirana stilska pravila u zasebnom dokumentu i može ga se referencirati s bilo koje HTML datoteke na web sjedištu (Goodman, 2002). Datoteka se mora održavati odvojena stilskim obrascem, a vanjski stilski obrazac nudi prednosti zbog lakšeg održavanja web stranice. Idealan je kada se stil primjenjuje na više stranica jer možemo promijeniti izgled cijelog web sjedišta promjenom samo jedne datoteke. Svaka stranica se povezuje korištenjem <link> elementa, koji se nalazi unutar <head> dijela HTML dokumenta (Harris, 2014). Vanjski stilski obrazac može biti napisan u bilo kojem uređivaču teksta, ali datoteka mora biti spremljena sa .css ekstenzijom (Boehrm i Ruvalcaba, 2022). Interni stilski obrazac nalazi se unutar same web stranice, a stilska pravila dodajemo u <style> element

zaglavlja dokumenta. Za određivanje formata teksta za određeni odlomak, riječ ili slovo koristi se umetnuti (engl. *inline*) stilski obrazac. Navedeni način je lošiji jer ga je moguće koristiti u samo određenom elementu (Robbins, 2018).

Slika 2 prikazuje CSS stil, u kojem se unutar elementa <body> nalazi svojstvo *background* koje određuje boju pozadine, a vrijednost je *teal*, odnosno plavozelena boja. Nakon njega nalazi se svojstva: *color*, koje definira boju teksta, *font-size*, koje određuje veličinu slova i *font-family*, koje određuje font. Sa svojstvom *margin-left* postavljamo lijevu marginu elementa, a osim navedenog margina može biti gornja, desna i donja (*margin-top*, *margin-right* i *margin-bottom*). S lijeve strane prikazan je navedeni stil, a s desne izgled navedene web stranice.

```
<!DOCTYPE html>
<html lang="hr">
 <head>
    <title>Naslov dokumenta</title>
  </head>

  <style>

  body {
    background: teal;
    color: yellow;
    font-size: 18px;
    font-family: Garamond;
    margin-left: 72px;
    }
  </style>
 <body>
   <h1>Glavni naslov</h1>
   <h2>Podnaslov</h2>

   <p>Ovo je paragraf nekog teksta</p>

   <img src="vir.jpg" alt="Vir" width="200" height="250">

  </body>
</html>
```



Slika 2. *Prikaz CSS dokumenta 2.0*

2.3. JavaScript

JavaScript je programski jezik koji podržava objektno orijentiranu paradigmu. Objektno orijentirano programiranje (OOP) je model računalnog programiranja kod kojeg se dizajn, donosno arhitektura temelji na podacima ili objektima, a ne funkciji. Jezik je jednostavan za korištenje i najčešće se koristi kao komponenta web stranica. Inicijalno je nazvan LiveScript, ali je Netscape promijenio u ime JavaScript, te se prvi put pojavio 1995. godine. Specifikacija ECMA - 262 definirala je standardnu verziju JavaScript jezika kao laganog, dizajniranog za stvaranje aplikacija usmjerenih na mrežu, komplementarnog s Javom i HTML-om. Upotreba

JavaScripta proširila se na razvoj mobilnih aplikacija, aplikacija za stolna računala i razvoj igara. Kod se izvršava na neku akciju, kao npr. na klik i fokus, na učitavanje dokumenta ili na okidač (Delamater i Ruvalcaba, 2020).

Za korištenje JavaScript koda, skripta mora biti uključena ili referencirana iz HTML dokumenta, kako bi se kod interpretirao putem web preglednika. JavaScript kod nalazi se unutar skupa oznaka <script> </script> jer početna oznaka <script> ima jedan obavezan atribut i jedan izborni atribut. Obavezni atribut je *type*, a neobavezni *src* koji definira URL datoteke vanjske skripte (McGrath, 2020). U Kodu 5 se programski kod nalazi između oznake otvaranja i zatvaranja skripte, te je prikazana vrijednost atributa text/javascript i standardna JavaScript funkcija prikazivanja teksta između navodnika. Datoteka ima nastavak .js.

```
1  <script type="text/javascript">
2    document.write("Neki tekst za završni rad");
3  </script>
```

Kod 5. *Prikaz JavaScript dokumenta 1.0*

Datoteku se može referencirati iz HTML koda putem *src* atributa, kao što je prikazano u kodu 6.

```
1  <script type="text/javascript" src="write.js"></script>
```

Kod 6. *Prikaz JavaScript dokumenta 2.0*

Navedeni postupak ima isti učinak kao i pisanje koda između oznake skripte, ali neće opteretiti HTML kod JavaScriptom, a dodatna je prednost što se ista skripta može uključiti u više stranica. Varijable su spremnici za pohranjivanje podatka u JavaScriptu i deklariraju se s ključnom riječi *var*. Varijable koje su deklarirane unutar funkcije nazivaju se lokalne, a varijable deklarirane izvan neke funkcije nazivaju se globalne varijable. Program može imati isto ime za lokalne i globalne varijable, ali vrijednost lokalne varijable unutar funkcije će imati prednost (Robbins, 2018). Primjer varijable prikazan je u kod 7.

```
var x = "Lorena";
```

Kod 7. *Prikaz JavaScript dokumenta 3.0*

JavaScript ima većinu operatora koji se također koriste u programskim jezicima Java/C, odnosno aritmetičke i logičke operatore, operator za usporedbu, operator dodjele te operatore nad bitovima.

Funkcija je segment koda za višekratnu upotrebu koja se može pozvati bilo gdje u JavaScriptu, što eliminira potrebu pisanja istog koda, u slučaju da se na više mjesta zahtijeva ista funkcionalnost. Neke od već ugrađenih funkcija u JavaScript su alert() i write(). Funkcije u JavaScriptu se deklariraju ključom riječi *function*. Primjer deklariranja funkcije Hello prikazan je u Kod 8.

```
<script type = "text/javascript">
  <!--
    function Hello() {
      alert("Pozdrav");
}
//-->
</script>
```

Kod 8. *Prikaz JavaScript dokumenta 4.0*

Komentari u JavaScriptu se označavaju sa // i /* *\, kao i u Java i C++. Objekti se sastoje od atributa. Ako atribut sadrži funkciju smatra se da je on metoda objekta, dok se u suprotnome atribut smatra svojstvom. Neki od objekata koji su već ugrađeni u JavaScript su: Boolean, String, Date, Array, Math. Boolean objekt predstavlja dvije vrijednosti, odnosno *true* i *false*. U Kodu 9. prikazan je primjer deklaracije i inicijalizacije objekta tipa Boolean.

```
var val = new Boolean(value);
```

Kod 9. *Prikaz JavaScript dokumenta 5.0*

Objekt tipa String koristi se za podatke koji su predstavljeni u tekstualnom obliku, a sam tekst se piše unutar navodnika (McGrath, 2020). String se eksplicitno može izraditi korištenjem konstruktora, kao što je prikazano u Kodu 10.

```
var sObject = new String("Napisan je neki niz");
```

Kod 10. *Prikaz JavaScript dokumenta 6.0*

Objekt tipa Date koristi se za rad s instancama datuma, a sadrži nekoliko metoda za pristupanje ili mijenjanje datuma ili njegovih komponenti. Instanciranje datuma bez prosljeđivanja parametra generira datum na temelju sata na računalu klijenta, primjer je prikazan u Kod 11.

```
var dtNow = new Date();
```

Kod 11. *Prikaz JavaScript dokumenta 7.0*

Objekt tipa Array omogućuje pohranjivanje više vrijednosti u jednu varijablu. U Kodu 12. prikazana je sintaksa kreiranja Array objekta.

```
var fruits = new Array( "apple", "orange", "mango" );
```

Kod 12. *Prikaz JavaScript dokumenta 8.0*

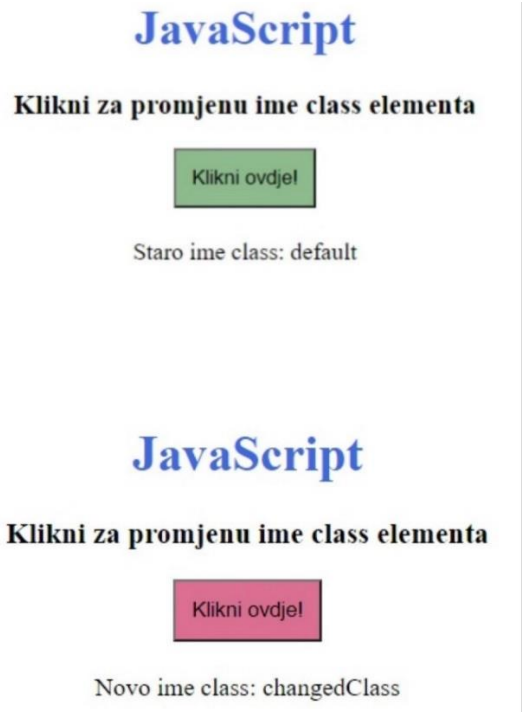
Klasa Math pruža svojstva i metode za matematičke konstante i funkcije. Sva svojstva i metode su statični, što znači da za njihovo korištenje nije potrebna inicijalizacija. Na primjer, konstantu PI je moguće dobiti s Math.PI, a sinusnu funkciju s Math.sin(x), gdje je x argument metode (Delamater i Ruvalcaba, 2020). Primjer pozivanja konstante PI i funkcije sin prikazan je u Kodu 13.

```
var pi_val = Math.PI;
var sine_val = Math.sin(30);
```

Kod 13. *Prikaz JavaScript dokumenta 9.0*

JavaScript također omogućava interaktivnost, poput pomicanja, klikanja, promjene stila unutar HTML i CSS dokumenta, kao što je prikazano na Slici 3. S lijeve strane nalazi se kod, a s desne strane izgled i funkcionalnost. Gornja desna slika prikazuje prvo što se prikaže kada se stranica otvori u web pregledniku, a druga slika kada se klikne na gumb „Klikni ovdje“. Pritom se promjeni vrijednost atributa *class* elementa s identifikatorom *myButton*.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Mijenjanje elementa class sa javascript</title>
        <style type="text/css">
            .default{
                background-color: darkseagreen;
            }
            .changedClass{
                background-color: palevioletred;
            }
            #myPara{
                margin-top: 20px;
            }
            #myButton{
                padding: 10px;
            }
            body {
                text-align:center;
            }
            h1 {
                color: royalblue;
            }
        </style>
        <script type="text/javascript">
            function changeClass() {
                document.getElementById('myButton').className =
                "changedClass";
                var button_class =
                document.getElementById('myButton').className;
                document.getElementById('myPara').innerHTML = "Novo ime
                class: "
                + button_class;
            }
        </script>
    </head>
    <body>
        <h1>JavaScript</h1>
        <h3>Klikni za promjenu ime class elementa</h3>
        <button class="default" onclick="changeClass()"
                id="myButton">Klikni ovdje!</button><br>
        <p id="myPara">Staro ime class: default</p>

    </body>
</html>
```



Slika 3. *Prikaz JavaScript 3.0*

3. Framework

Razvojni okvir (engl. *framework*) je platforma koja pruža temelj za razvoj softverskih aplikacija. Zamišljen je kao predložak koji se može selektivno mijenjati dodavanjem koda, koji koristi zajedničke resurse, npr. slike, dokumente i knjižnice, i koji ih spaja u jedan paket. Taj se paket može modificirati kako bi odgovarao specifičnim potrebama projekta. U razvojne okvire, razvojni inženjeri mogu dodati ili zamijeniti značajke kako bi aplikaciji dali novu funkcionalnost. Razvijen je veliki broj dostupnih razvojnih okvira, koji su namijenjeni rješavanju različitih problema koji se pojavljuju u procesu razvoja softvera. Backend web razvojni okvir pomaže razvojnim inženjerima u izradi web aplikacija i dinamičkih web stranica. Razvojni okviri su transformirali način izrade weba, ubrzavajući cijeli proces softvera automatizacijom zadataka za web programe, poput pružanja pristupa bazama podataka, upravljanja sesijama i predložaka stranica (Metzgar, 2017).

Umjesto izrade web stranica osnovnim korištenjem HTML-a, CSS-a i JavaScripta, razvojni okviri koriste programske jezike za interakciju s bazom podataka kako bi generirali sadržaj. Najpoznatiji backend razvojni okviri su Django (Python) i Rails (Ruby) (Metzgar, 2017). Frontend razvojni okvir razvojnim inženjerima omogućuje definiranje dizajna, odnosno izgleda web stranice, kao što je upravljanje AJAX zahtjevima, definiranje strukture datoteka i stiliziranje komponenti web stranice. AngularJS (JavaScript), React (JavaScript) i Bootstrap (CSS) su najčešći frontend razvojni okviri (Metzgar, 2017). Razvojni okviri mobilnih aplikacija pružaju razvojnim inženjerima strukturu koja podržava proces izgradnje mobilnih aplikacija. Uobičajeni razvojni okviri su Flutter i React Native. Razvojni okvir smanjuje vrijeme i energiju utrošenu na razvoj softvera jer pruža generički radni sustav koji korisnik može razviti za određenu aplikaciju proširenjem koda.

JavaScript razvojni okviri bitan su dio modernog frontend web razvoja, pružajući razvojnim inženjerima isprobane i testirane alate za izgradnju skalabilnih i interaktivnih web aplikacija. Iako se razvojni okviri i biblioteke ponekad koriste kao istoznačnice, postoji važna razlika među njima. JavaScript razvojni okvir je potpuni skup alata koji pomaže u oblikovanju i organiziranju web stranice ili aplikacije, dok je JavaScript biblioteka zbirka unaprijed napisanih isječaka koda koji se u manjoj mjeri fokusiraju na oblikovanje aplikacije. Najpoznatije JavaScript biblioteke su jQuery i React library. Moderni JavaScript razvojni okviri koriste model temeljen na Model-View-Controller (MVC) oblikovnom obrascu. MCV koristi se za razvoj korisničkih sučelja koja dijele srodnu programsku logiku u tri međusobno povezana elementa. Komponenta Model odgovara cijeloj logici vezanoj uz podatke,

komponenta View koristi se za svu logiku korisničkog sučelja aplikacije, dok Controller djeluje kao sučelje između komponenti Model i View. Conntroller obrađuje dolazne zahtjeve, koristi poslovnu logiku i manipuliranje podacima preko komponente Model i interakciju s komponentom Views kako bi prikazali konačni izlaz. Neki od najpoznatijih frontend okvira su React.js, AngularJs, Vue.js, Ember, te backend su Express.js, Next.js, Meteor.js itd. (Metzgar, 2017).

3.1. AngularJs

AngularJs je JavaScript razvojni okvir koji služi za izradu web, mobilnih aplikacija. Razvoj AngularJs aplikacija uključuje korištenje Typescripta, koji je nadskup JavaScripta, zajedno s HTML-om i CSS-om. Kod napisan u Typescriptu kompilira se u JavaScript i prikazuje u pregledniku. Izvorno ga je 2009. godine razvio Miško Hevery iz Brat Tech LLS kao softver iza online usluge za pohranu JSON-a za laku izradu aplikacija. JSON je tekstualni format za pohranu i prijenos podataka. Kao platforma, AngularJs uključuje komponentni razvojni okvir za izgradnju skalabilnih web aplikacija, zbirke dobro integriranih biblioteka koje pokrivaju širok raspon značajki (usmjeravanje, upravljanje obrascima, komunikaciju klijet – poslužitelj, itd.) te skup alata za razvojne inženjere koji će pomoći da razviju, izgrade, testiraju te ažuriraju kod (Seshadri i Green, 2013). AngularJs se distribuira kao JS datoteka i može se dodati na web stranicu pomoću oznake <script> kao što je prikazano u Kod 14.

```
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js">
</script>
```

Kod 14. *Prikaz AngularJs dokumenta 1.0*

AngularJs proširuje HTML sa ng-smjernicama. Direktiva ng-app definira AngularJs aplikaciju, ng-model veže vrijednost HTML kontrola (unos, odabir, tekstualno područje) na aplikacije, a ng-bind povezuje podatke aplikacije s HTML prikazom. AngularJs direktive su HTML atributi s prefiksom ng. Direktiva ng-init inicijalizira varijable aplikacije AngularJs, kako je prikazano u Kod 15.

```
<div ng-app="" ng-init="firstName='Lorena'">

<p>Ime je <span ng-bind="firstName"></span></p>

</div>
```

Kod 15. *Prikaz AngularJs dokumenta 2.0*

AngularJs izrazi (engl. *expressions*) su napisani unutar dvostrukih vitičastih zagrada: {{ neki izraz }}. AngularJs će uključiti, odnosno prikazati podatke na mjestu gdje je izraz napisan, kao što je prikazano u Kodu 16. Kao rezultat navedenog koda, ispisat će se, „Moj zbroj: 10“. AngularJs izrazi vežu podatke AngularJs za HTML na isti način kao ng-bind direktiva.

```
<!DOCTYPE html>
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"
></script>
<body>

<div ng-app="">
  <p>Moj zbroj: {{ 5 + 5 }}</p>
</div>

</body>
</html>
```

Kod 16. *Prikaz AngularJs dokumenta 3.0*

Promatrač (engl. *watcher*) pohranjuje funkciju koju koristimo da dobijemo najnoviju vrijednost stavke, posljednju poznatu vrijednost, te povratni poziv. Kada se pokrene ciklus sažetaka, Angular petlja prolazi kroz sve promatrače i izvršava prvu funkciju, koja vraća najnoviju vrijednost stavke. Zatim uspoređuje posljednju vrijednost i novu vrijednost, pozivajući povratni poziv kada su vrijednosti različite.

AngularJs modul (engl. *modules*) definira AngularJs aplikacije, AngularJs kontroler (engl. *controllers*) kontrolira AngularJs aplikaciju. Direktiva ng-app definira aplikaciju, a direktiva ng-controller definira kontroler. Moduli i kontroleri mogu se napraviti u istoj datoteci zajedno s HTML datotekom, a ako se želi koristiti modul u drugoj datoteci, treba se izraditi modul, kontroler i HTML datoteka zasebno (Karpov i Netto, 2015). Modul i kontroler treba pohraniti u datoteku s ekstenzijom .js. U kodu 17. prikazan je primjer AngularJs modul i AngularJs

kontrolera. Objekt „firstName“ i „lastName“ vezani su uz $scope i njegovim se metodama pristupa korištenjem izraza, ng-model.

```
<!DOCTYPE html>
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js">
</script>
<body>

<p>Promjena imena</p>

<div ng-app="myApp" ng-controller="myCtrl">

First Name: <input type="text" ng-model="firstName"><br>
Last Name: <input type="text" ng-model="lastName"><br>
<br>
Full Name: {{firstName + " " + lastName}}

</div>

<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.firstName= "Lorena";
    $scope.lastName= "Šubić";
});
</script>

</body>
</html>
```

Kod 17. *Prikaz AngularJs dokumenta 4.0*

AngularJs pruža kontrolnu uslugu naziva AJAX-$http, koja omogućuje komunikaciju s udaljenim poslužiteljima. Podaci su uglavnom potrebni u JSON formatu. Postoji nekoliko metoda koje se mogu koristiti za pozivanje usluge $http, a neke od njih su: .post, .get, .head, .jsonp, .patch, .delete te .put. Objekt je odgovor poslužitelja, uz pomoć svojstva. Primjer svojstva: .headers (dobivanje informacija zaglavlja), .statusText (definiranje HTTP statusa), .data (prijenos odgovora s poslužitelja), .config (generiranje zahtjeva). Sintaksa AJAX-$http prikazana je u Kod 18.

```
function studentController($scope,$https:) {
   var url = "data.txt";

   $https:.get(url).success( function(response) {
      $scope.students = response;
   });
}
```

Kod 18. *Prikaz AngularJs dokumenta 5.0*

AngularJs API se koristi za uspoređivanje, ponavljanje i pretvaranje objekata. API je softverski posrednik koji omogućuje da dvije aplikacije komuniciraju jedna s drugom (Seshadri i Green, 2013). Osnovni AngularJs API uključuje: angular.isString, koji provjerava je li objekt niz znakova, angular.lowercase, koji pretvara znakove niza u mala slova, angular.uppercase koji pretvara znakove niza u velika slova, angular.isNumber, koji provjerava je li objekt broj. Primjer provjere je li objekt broj i koji je odgovor prikazan je na slici 4.

Slika 4. *Prikaz AngularJS 4.0*

3.2. Vue.js

Vue.js je JavaScript razvojni okvir za izradu korisničkih sučelja. Temelji se na HTML-u, CSS-u i JavaScriptu, a pruža deklarativni i komponentni model programiranja koji ubrzava razvoj korisničkog sučelja. Kako bi se koristio Vue.js, potrebno ga je instalirati, a jedan od načina je korištenje CND (Content Delivery Network). Dvije ključne značajke Vue.js su deklarativno prikazivanje i reaktivnost. Kod deklarativnog prikazivanja Vue.js proširuje standardni HTML sintaksom predloška koja omogućuje deklarativno opisivanje HTML izlaza na temelju stanja JavaScripta. Kod reaktivnosti Vue.js automatski prati promjene stanja JavaScripta i učinkovito ažurira DOM (Document Object Model) kad se promjene dogode. Vue.js je dizajniran da bude fleksibilan i postepeno prilagodljiv. Vue.js se može koristiti za poboljšanje statičkog HTML-a bez koraka izrade, može se ugraditi kao web komponenta na

bilo koju web stranicu, aplikaciju za jednu stranicu, tzv. fullstack, odnosno renderiranje na strani poslužitelja i tzv. jamstack, odnosno generiranje statičke stranice (Macrae, 2018).
Vue.js komponente kreiramo koristeći format datoteke nalik HTML-u koji se naziva SFC, tj. komponente sadržane u jednoj datoteci (Single – File Component). SFC sadrži logiku komponente (JS), predložak (HTML) i stilove (CSS) u jednoj datoteci. Vue.js je fokusiran na sloj ViewModel uzorka MVVM (model-view-viewmodel). ViewModel je objekt koji sinkronizira View i Model (Djirdeh, Murray i Lerner, 2018). U kodu 19. prikazano je instanciranje s Vue.js konstruktorom.

```
var vm = new Vue({ /* options */ })
```

Kod 19. *Prikaz Vue.js 1.0*

Vue.js direktive su HTML atributi sa prefiksom v-. Vue.js koristi dvostruke vitičaste zagrade {{ }} kao držač mjesta za podatke. Vue.js ima ugrađene direktive kao što su v-if, v-else, v-show, v-on, v-bind i v-model koje se koriste za izvođenje različitih radnji na frontendu. Osim zadanog skupa direktiva koje se isporučuje u jezgri, Vue.js također omogućuje registraciju vlastitih prilagođenih direktiva. U kodu 20. prikazano je korištenje v-if direktive. Direktiva v-if će proći samo ako je jedna od tvrdnji točna. U kodu su ponuđene tri direktive, za svako vremensko razdoblje jedna.

```
<!DOCTYPE html>
<html>
<script src="https://cdn.jsdelivr.net/npm/vue@2.6.14/dist/vue.js"></script>
<body>

<h2>Zavrsni</h2>

<div id="app">
  <p v-if="isMorning">Good morning!</p>
  <p v-if="isAfternoon">Good afternoon!</p>
  <p v-if="isEvening">Good evening!</p>
</div>
<script>
var hours = new Date().getHours();

new Vue({
  el: '#app',
  data: {
      isMorning: hours < 12,
      isAfternoon: hours >= 12 && hours < 18,
      isEvening: hours >= 18
    }
  });
</script>
```

Kod 20. *Prikaz Vue.js 2.0*

Vue.js koristi sintaksu predloška temeljenu na HTML-u koja omogućuje deklarativno povezivanje renderiranog DOM-a s podacima instance temeljne komponente. View je stvarni

DOM kojim upravljaju Vue.js instance. Vue.js koristi šablone temeljene na DOM-u, te svaka instanca je povezana s odgovarajućim DOM elementom. Kada se kreira instanca, ona rekurzivno prolazi po svim podređenim čvorovima svog korijenskog elementa dok postavlja potrebna vezanja podataka. Nakon što je View kompiliran, postaje aktivan promjenom podataka (Djirdeh, Murray i Lerner, 2018).

U Vue.js Modeli su JavaScript objekti ili objekti podataka. Kada se objekt koristi kao podatak unutar Vue.js instance, on postaje reaktivan, tj. može se manipulirati svojstvima. Komponente Vuea mogu se izraditi u dva različita API stila: Options API (hrv. *opcije*) i Composition API (hrv. *kompozicija*). API opcija definira logiku komponenata koristeći objekt opcija kao što su *data, methods* i *mounted.* Svojstva definirana opcijama izložena su sa *this* unutar funkcije. S kompozicijom API definira se logika komponente, koristeći uvezene API funkcije. U SFC, kompozicija API koristi se s <script setup>. U Vue.js značajka vezanja podataka pomaže manipulirati ili dodijeliti vrijednosti HTML atributima, promijeniti stil, dodijeliti klase uz pomoć direktive vezanja pod nazivom v-bind (Macrae, 2018). U kodu 21. prikazana je sintaksa direktive v-bind.

```
v-bind:attribute="expression";
```

Kod 21. *Prikaz Vue.js 3.0*

Za dvosmjerno vezanje u Vue.js koristi se direktiva v-model, koja veže vrijednost HTML elementa za podatke aplikacije. U kodu 22. prikazano je korištenje v-model uz modifikator .number, koji se koristi za automatski prikaz broja od strane korisnika.

```
<input v-model.number="age" />
```

Kod 22. *Prikaz Vue.js 4.0*

V-show je Vue.js direktiva koja se koristi za prebacivanje prikaza CSS svojstva elemenata sa zadanim podacima. Ako su podaci istiniti, učinit će ih vidljivima, a u suprotnom će ih učinit nevidljivima. Sintaksa v-show prikazana je u Kodu 23.

```
v-show="data"
```

Kod 23. *Prikaz Vue.js 5.0*

Direktiva v-on:click koristi se za definiranje funkcionalnosti koja će se izvršiti nakon klika na element. U kodu 24. prikazana je sintaksa.

```
v-on:click="function"
```

Kod 24. *Prikaz Vue.js 6.0*

Filter je jednostavna JavaScript funkcija koja se koristi za promjenu izlaza podataka u web pregledniku. Filteri u Vue.js ne mijenjaju podatke izravno gdje su pohranjeni, već samo primjenjuje formatiranje podataka. Vue.js ne daje filtere prema zadanim postavkama, nego se moraju kreirati. Filteri mogu biti globalni i lokalni. Globalni filter omogućuje pristup svim komponentama, dok lokalni filter djeluje unutar komponenata u kojima je definiran (Djirdeh, Murray i Lerner, 2018). Na slici prikazan je primjer koda globalnog filtra (lijevo) i izgled stranice u web pregledniku (desno).

```
<!DOCTYPE html>
<html>
<script src="https://cdn.jsdelivr.net/npm/vue@2.6.14/dist/vue.js"></script>
  <body>

<h2>Vue završni</h2>

  <div id="app">
    <p>Cijena prvog produkta: {{ productOneCost | formatCost }}</p>
    <p>Cijena drugog produkta: {{ productTwoCost | formatCost }}</p>
    <p>Cijena trećeg produkta: {{ productThreeCost | formatCost }}</p>
  </div>
<script>
  new Vue({
    el: '#app',
    data: {
      productOneCost: 234,
      productTwoCost: 4567,
      productThreeCost: 78910
    },
    filters: {
      formatCost(value) {
        return '€' + (value / 100).toFixed(2);
      }
    }
  });
</script>
  </body>
</html>
```

Vue završni

Cijena prvog produkta: €2.34

Cijena drugog produkta: €45.67

Cijena trećeg produkta: €789.10

Slika 5. *Prikaz Vue.js 5.0*

3.3. React.js

React.js popularna je JavaScript biblioteka otvorenog koda koja prikazuje podatke uz pomoć HTML-a. React.js nova je tehnologija u usporedbi s drugim tehnologijama na tržištu. Nastala je 2011. godine, a kreirao ju je Jordan Walke, softverski inženjer u Facebooku. React.js je pod utjecajem XHP-a koji je jednostavan okvir HTML komponente za PHP. React.js se prvi

put ozbiljnije koristio za razvoj Facebooka, a kasnije se koristio za razvoj komponenti Instagrama. S vremenom je React.js rastao, a Facebook ga je odlučio učiniti otvorenim kodom 2013. godine. Godine 2015. najavljen je JSConf, React Native koji je omogućio jednostavan razvoj za Android i IOS. Iste godine je React.js Native postao softver otvorenog koda (Eisenman, 2017).

React Native je okvir otvorenog koda koji se koristi za izgradnju bogatog mobilnog korisničkog sučelja od deklarativnih komponenti koristeći samo JavaScript te omogućuje razvoj mobilnih aplikacija s jednom bazom koda. React Native koristi osnovnu apstrakciju React.js, a samo su komponente knjižice različite. Dok React.js koristi Virtual DOM, React Native koristi API-je za renderiranje komponenti koje se mogu ponovno koristiti za Android i iOS platforme. React Native, za razliku od React.js, ne koristi CSS i HTML.

React.js nudi neke značajke koje ga čine najprihvaćenijom bibliotekom za razvoj frontend aplikacija. JSX je React ekstenzija koja omogućuje pisanje JavaScript koda koji izgleda kao HTML. JSX je sintaksa slična HTML-u u koju koristi React koja proširuje ECMAScript tako da sintaksa slična HTML-u može koegzistirati s JavaScript/React kodom. ECMAScript (ES) je programski jezik opće namjene koji je implementiran u JavaScriptu. JSX omogućuje pisanje struktura u istoj datoteci u kojoj se piše JavaScript kod, a zatim transformira izraze u stvarni JavaScript kod. Kao i XML/HTML, JSX oznake imaju naziv oznake, atribut i djecu. U kodu 25. prikazano je kako je JSX implementiran u React.js. ugrađuje HTML u JavaScript kod.

```
const name = 'Simplilearn';
const greet = <h1>Zdravo, {name}</h1>;
```

Kod 25. *Prikaz React.js 1.0*

Virtualni DOM je Reactova lagana verzija Real DOM-a. Manipulacija stvarnim DOM-om znatno je sporija od manipulacije virtualnim DOM-om. Kada se stanje objekta promijeni, virtualni DOM ažurira samo taj objekt u stvarnom DOM-u, a ne sve. To je jedan od razloga zašto se radnje odvijaju brže u usporedbi s drugim frontend tehnologijama koje moraju ažurirati svaki objekt čak i ako se promijeni samo jedan objekt u web aplikaciji. (Eisenman, 2017).
React.js je dizajniran na način da prati jednosmjerni tok podataka ili jednosmjerno vezanje podataka. Prednosti jednosmjernog povezivanja podataka daju bolju kontrolu u cijeloj aplikaciji.

React.js se temelji na komponentama koje su navedene kao prilagođene HTML oznake, zbog čega je jednostavno koristiti. React.js je prilično napredan u zaštiti unutarnjih komponenti i protoka podataka. Na podređene komponente se ne može izravno utjecati vanjskim upitima, što je prednost prilikom razvoja prikaza u frontendu. React.js dijeli korisničko sučelje na brojne komponente, a svaka komponenta ima vlastiti skup svojstava i funkcija. Neke od značajki komponenti su:

- ponovno korištenje - komponenta koja se koristi u jednom području aplikacije može se ponovno koristiti u drugom području,
- ugniježđene komponente - komponenta može sadržavati nekoliko drugih komponenti
- metoda renderiranja - u svom minimalnom obliku, komponenta mora definirati metodu renderiranja koja definira način renderiranja komponente u DOM-u, itd.

Vrlo je učinkovit u ažuriranju HTML dokumenata kada se promijene vrijednosti podataka, što ga čini dobrim izborom za web aplikacije koje se temelje na podacima kao što su Facebook i Instagram (Stefanov, 2021).

Rekviziti (engl. *props*) se koriste za slanje podataka i rukovatelja događajima podređenim komponentama, pri čemu nadređena komponenta postavlja rekvizite za podređene komponente. Nakon što su jednom postavljeni, ne mogu se mijenjati. Kod 26. prikazuje primjer komponente koja prima argument kao *props* objekt.

```
function Car(props) {
  return <h2>I am a { props.brand }!</h2>;
}
```

Kod 26. *Prikaz React.js 2.0*

React.js komponente imaju ugrađeni *state* objekt. Objekt *state* je mjesto gdje se pohranjuju vrijednosti svojstava koja pripadaju komponenti. Kada se objekt *state* promijeni, komponenta se ponovno prikazuje, što znači da će se izlaz promijeniti u skladu s novim vrijednostima. Objekt može sadržavati proizvoljan broj svojstava, kao što je prikazano u kodu 27., u koji je dodan brand, model i godina auta.

```
class Car extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      brand: "Opel",
      model: "Kadett",
      color: "yellow",
      year: 1979
    };
  }
  render() {
    return (
      <div>
        <h1>My Car</h1>
      </div>
    );
  }
}
```

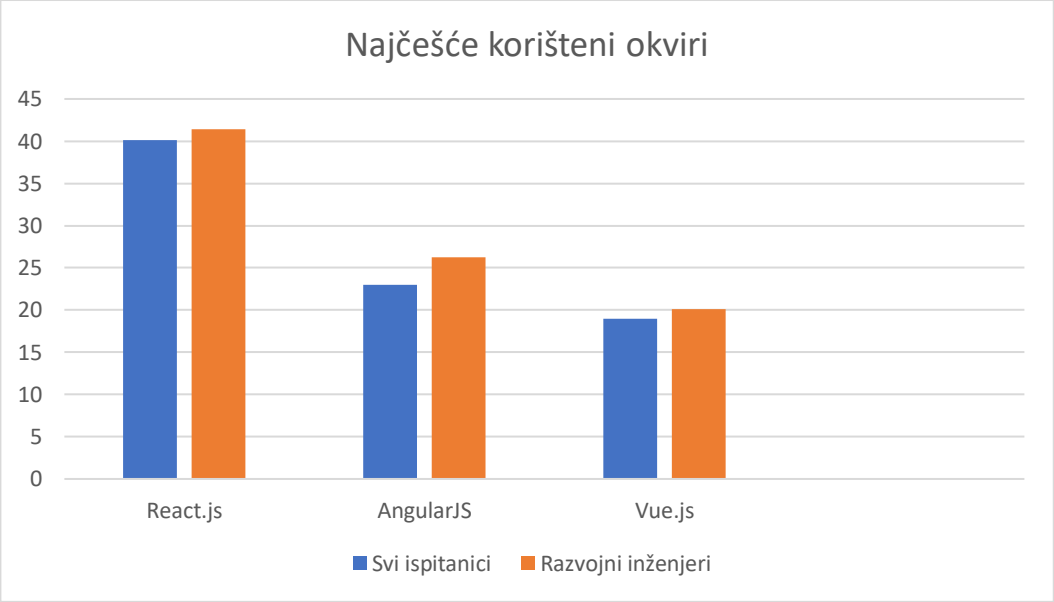
Kod 27. *Prikaz React.js 3.0*

Konstruktor (engl. *Constructor*) je metoda koja se koristi za inicijalizaciju stanja objekta u klasi, odnosno za inicijalizaciju lokalnog stanja komponente i za vezanje metoda rukovatelja događajima koje se pojavljuju u komponenti (Stefanov, 2021). Automatski se poziva tijekom stvaranja objekta u klasi. Konstruktor u komponenti mora pozvati metodu *super(props)* prije bilo koje druge metode. Ako komponenta treba pristup lokanom stanju, mora se koristiti *this.state* za dodjelu početnog stanja u konstruktoru. Konstruktor koristi *this.state* samo za dodjelu početnog stanja, a za sve ostale metode koristi metodu *set.state()*.

3.4. Statistika

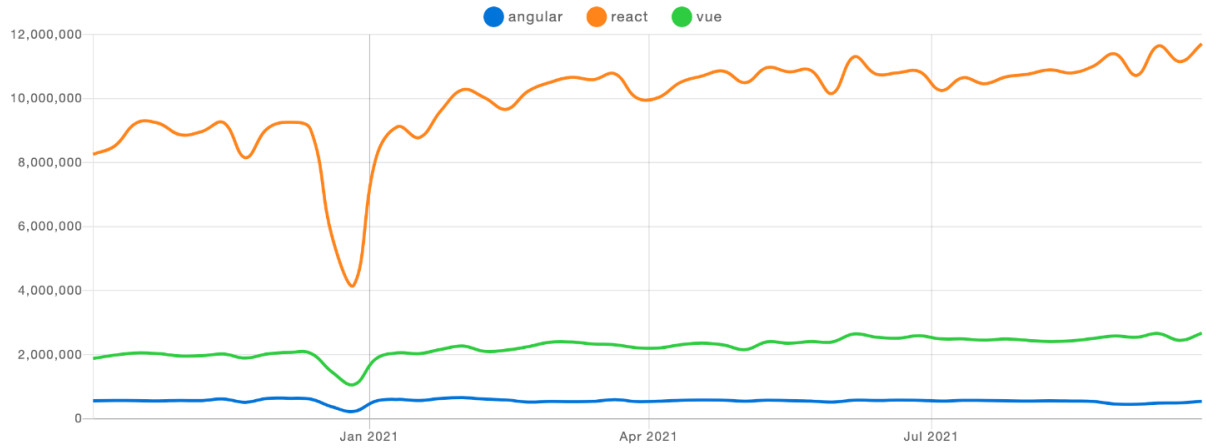
Provedena istraživanja daju nam uvid u učestalost korištenja pojedinih web okvira, preference korisnika prilikom odabira okvira, te broj preuzimanja i ocjenu pojedinih okvira. U ovom poglavlju će se prikazati podaci za tri najkorištenija okvira, AngularJS, React.js te Vue.js, i usporedit će se podaci dobiveni u istraživanjima Stack Overflowa koje ispituje koji su najkorišteniji web okviri u skupini profesionalnih razvojnih inženjera, te općenito populacije koja se koristi tim web okvirima. Nakon toga prikazat će se NPM trendovi preuzimanja svakog od navedenih web okvira te GitHub zvjezdice koje su mjerilo popularnosti projekta otvorenog koda. Također će se usporediti okviri prema tržištu rada, te kategorije web stranica koje su najviše korištene među okvirima.

U istraživanju „Stack Overflowa“ je sudjelovalo 67,593 ispitanika. Podaci ukazuju na to kako je React.js najčešće korišten web okvir među 40,1% ispitanika, a AngularJS među 22,9% ispitanika, dok je Vue.js najkorišteniji među 18,9% ispitanika. U poduzorku razvojnih inženjera React.js se pokazao kao najkorišteniji okvir kod 41,4% ispitanika, poslije njega, od okvira koje analiziramo u radu, najkorišteniji je AngularJS sa 26,23%. Na posljednjem mjestu nalazi se Vue.js sa 20,09%, kao što je vidljivo u grafu 1. Usporedivši podatke između svih ispitanika i razvojnih inženjera vidljivo je kako su svi navedeni okviri u subuzorku razvojnih inženjera zastupljeniji nego li je to slučaj u ukupnom uzorku. Ti nam podaci govore da su ovi web okviri vrlo popularni i korišteni, no još su zastupljeniji u uzorku razvojnih inženjera.



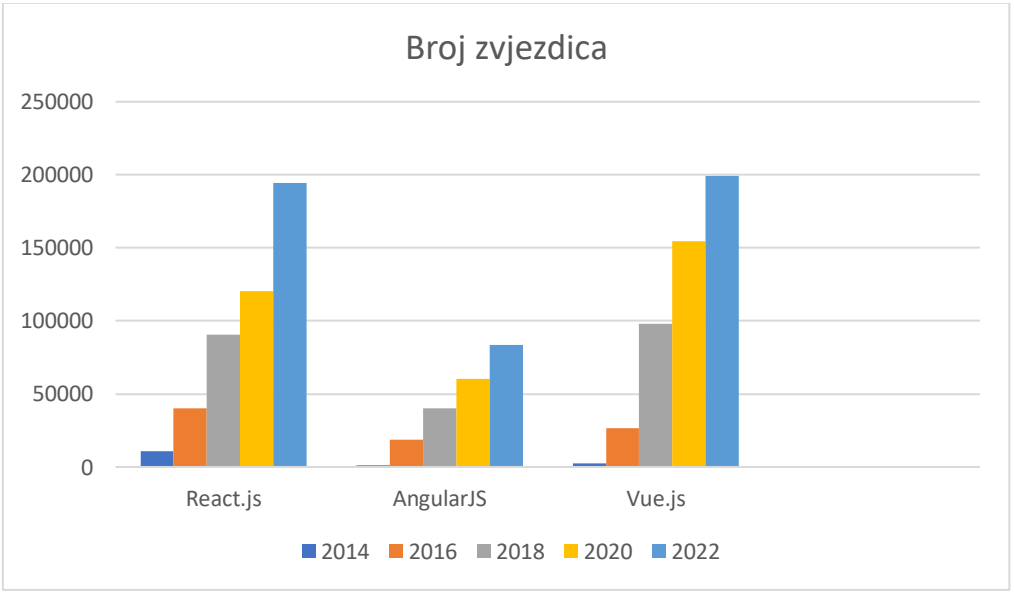
Graf 1. *Usporedna najčešće korištenih okvira između svih ispitanika i razvojnih inženjera (Preuzeto sa Vue vs React vs Angular: What Framework Would You Choose? (2020))*

NPM trendovi procijenili su koliko je puta svaki od navedenih okvira (AngularJS, React.js te Vue.js) preuzet za web razvoj u razdoblju od 2021. do 2022. godine. Kao što je prikazano u grafu 2. vidljivo je da React.js ostaje na vodećoj poziciji, što znači da je najčešće preuziman, a za njim slijedi Vue.js, dok je na mjestu najmanje preuzimanog okvira AngularJS. Uzevši u obzir graf 1. koji prikazuje frekvenciju korištenja i graf 2. koji prikazuje broj preuzimanja, vidljivo je da je React.js vodeći u oba slučaja. Nakon njega se nalazi najmlađi od okvira, Vue.js, a od proučavanih najmanje je preuziman i korišten AngularJS.



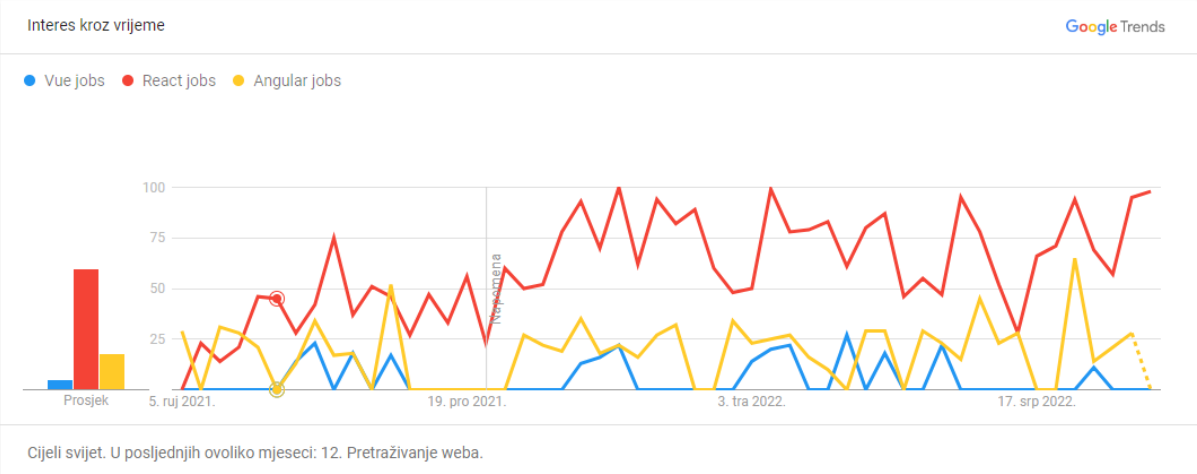
Graf 2. *Preuzimanje okvira u godini dana, NPM (Preuzeto sa Angular Vs React Vs Vue: The Main Differences And Use Cases (2022))*

Dobar pokazatelj popularnosti također je i broj dobivenih zvjezdica na GitHub repozitoriju. GitHub zvjezdice su podaci koji pokazuju popularnost projekta otvorenog koda. U grafu 3. prikazana je nagla promjena u broju zvijezda Vue.js koja se dogodila sredinom 2016., te je odnedavno Vue.js s Reactom.js među najpopularnijim okvirima. React.js je od početka 2014. imao najviše zvjezdica, dok ga Vue.js nije pretekao 2022. godine. Na posljednjem mjestu nalazi se AngularJS koji se u broj zvjezdica značajno zaostaje za Vue.js i React.js.



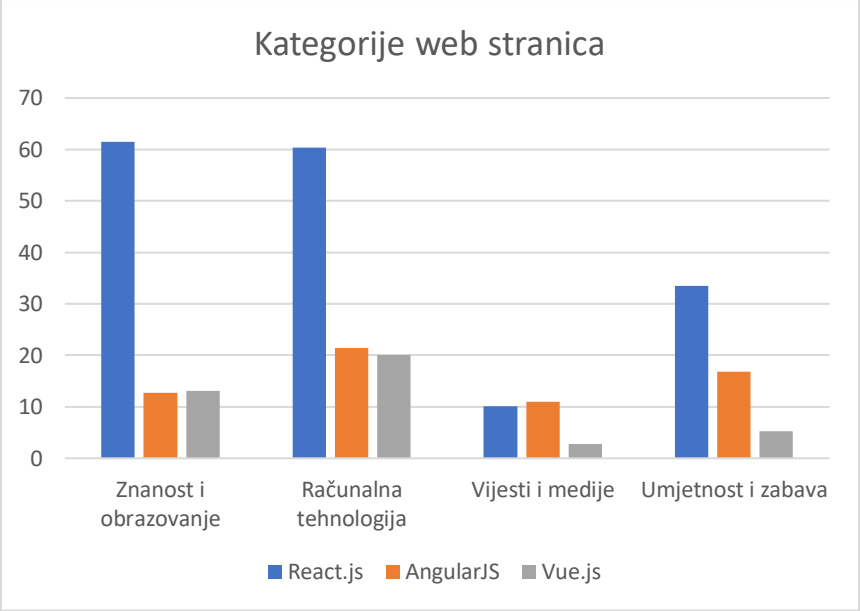
Graf 3. *Broj zvjezdica na GitHub projektima za AngularJS, React.js i Vue.js (Preuzeto sa Star History (2022))*

Broj poslova koji zahtijevaju skup vještina za AngularJS, React.js i Vue.js na tržišute rada u posljednjih 12 mjeseci (2021-2022) prema Google Trends prikazan je u grafu 4. Crvena boja predstavlja React.js koji je vodeći cijele godine, žuta boja AngularJS, dok plava predstavlja Vue.js. Sa stajališta trenutnog tržišta rada, najviše korišteni su React.js i AngularJS, ali je također vidljivo da se Vue.js ne razlikuje previše od AngularJS.



Graf 4. *Tržište rada prema Google Trends (Preuzeto sa Google Trends (2022))*

Graf 5. prikazuje učestalost pojedinih okvira za izradu različitih kategorija web stranica. Kao kategorije korištene su: Znanost i obrazovanje, Računalna tehnologija i Umjetnost i zabava, u kojima je React.js najkorišteniji. Dok su u kategoriji Vijesti i mediji podjednako zastupljeni React.js i AngularJS s blagim vodstvom AngularJS-a. Vue.js se podjednako koristi u različitim kategorijama, no nije vodeći okvir niti za jednu od navedenih kategorija web stranica.



Graf 5. *Korištenje kategorije web stranica između React.js, AngularJS i Vue.js (Preuzeto sa AngularJS vs Vue.js (2021))*

Neki od najpoznatijih usluga i aplikacija, odnosno Instagram, Whatsapp, Facebook, Airbnb, Uber, Netflix, Dropbox koriste React.js. S druge strane AngularJS koristi se za Google, Nike, General Motors, Upwork, Sony itd. Facebook i Instagram, uz React.js koriste, i Vue.js, a Vue.js omogućio je i korištenje Adobea, Xiaomia, Trivagoa, Grammarlya, Gitlaba i drugih.

Pretraživanje na *Indeed* 2022. pokazalo je da razvojni inženjeri koji koriste React.js imaju 67.301 poslovnu ponudu, oni koji koriste AngularJS imaju nešto manje (24.508 ponuda), dok najmanje poslovnih prilika imaju korisnici Vue.js-a (sa samo 3.857). Nedvojbeno je da je uz znanje React.js-a najlakše pronaći posao te da je on već dugi niz godina popularan među razvojnim inženjerima. Također može se svjedočiti jačanju opozicije Vue.js protiv Reacta.js u pogledu privrženosti zajednice Vue.js-u. AngularJS također zadržava svoju stabilnu srednju poziciju dobivajući pozitivne povratne informacije i od razvojnih inženjera i od poslodavaca.

3.5. Usporedba

Angular.js je frontend razvojni okvir i može se koristiti s bilo kojim pozadinskim programskim jezikom kao što su PHP, Java itd. Vue.js strogo temeljen za frontend i koristi HTML, CSS i JavaScript zasebno. React.js je JavaScript biblioteka koji se koristi za izgradnju UI okvira. Angular.js se ne mora posebno instalirati i može ga se dodati kao i svaku drugu JavaScript datoteku te je u potpunosti implementiran pomoću JavaScripta, Vue.js koristi CLI ili CND za

instalaciju sintakse predloška temeljenu na HTML-u i koncepte kao što su modeli i komponente dok su za React.js potrebne NodeJS i NMP platforme za razvoj aplikacije.

Angular.js je razvojni okvir otvorenog koda za klijentsku stranu aplikacije i podržava aplikacije u stvarnom vremenu kao što su aplikacije za razmjenu trenutnih poruka ili chata i implementira MVVM uzorak, dok je Vue.js otvoreni i progresivni razvojni okvir za izgradnju korisničkih sučelja i najprikladniji za manje aplikacije na jednoj stranici, pružajući jednostavno sučelje koji se fokusira na ViewModel, React.js koristi se za razvoj aplikacija i korisničkih sučelja.

Sintaksa Vue.js je u nekim elementima slična AngularJS (npr. v-if i ng-if) zato što je AngularJS napravio puno stvari i one su bile inspiracija za Vue.js u njegovom razvoju, no Vue.js ima jasniju razliku između direktiva i komponenti. Smjernice su namijenjene samo za DOM manipulaciju, dok su komponente samostalne jedinice koje imaju svoj pogled i logiku podataka. U AngularJS, direktive rade sve, a komponente su samo specifična vrsta direktive. JSX je sintaksa koju React.js koristi jer je brži od uobičajenog JavaScripta jer provodi optimizaciju dok prevodi kod u JavaScript.

AngularJS koristi dvosmjerno povezivanje između opsega (opseg koji se ne nasljeđuje od nadređenog, te je samostalan), dok se Vue.js temelji na jednosmjernom protoku podataka između komponenti. AngularJS postaje spor kada ima puno "promatrača", jer svaki put kada se nešto promijeni u opsegu, sve te "promatrače" treba ponovno procijeniti. Vue.js je u tom segmentu bolji jer koristi transparentni sustav praćenja ovisnosti s asinkronim redovima čekanja što znači da se sve promjene pokreću neovisno, osim ako nemaju eksplicitne odnose ovisnosti. React.js kao i Vue.js koristi jednosmjerni protok podataka što čini protok podataka jednostavnijim. Razlika između AngularJS i React.js tehnologije je što se AngularJS može u potpunosti testirati samo s jednim alatom.

Uzimajući u obzir sve prednosti i nedostatke, AngularJS je najbolji izbor za dinamične projekte koji koriste većinu značajki razvojnih okvira. Vue.js je bolji za manje projekte i druge aplikacije koje preferiraju veću brzinu nego veću fleksibilnost, te React.js najbolji je za razvoj korisničkih sučelja weba kao mobilna aplikacija. Postoji nekoliko stvari koje Vue.js i AngularJS imaju zajedničko. Oboje imaju koncept predložaka, komponenti, itd. Što se tiče značajki, gotovo su identični, međutim mala je razlika među njima, ali AngularJS protiv Vue.js, oba JavaScript razvojna okvira, mogu dati najbolje rezultate za razvoj aplikacije. React.js i Vue.js oboje koriste Virtual DOM model, te su oboje JavaScript okviri otvorenog koda.

AngularJS i React.js oboje imaju arhitekturu temeljenu za komponentama, što znači da imaju ponovno upotrebljive i modularne komponente.

4. Zaključak

U radu su prikazane komponente aplikacije rađene u klijent-server arhitekturi, a poseban naglasak je stavljen na razvoj klijentskog dijela web aplikacije. Objašnjen je klijentski dio web aplikacije i web stranice koristeći se tehnologijama poput HTML-a, CSS-a i JavaScripta.

HTML je jednostavan prezentacijski jezik pomoću kojeg se može napraviti statična web stranica. Web stranica može biti jednostavna, te se na njoj mogu nalaziti osnove web stranice kao što su naslovi, odlomci, slika, tablice itd. Pomoću HTML-a se ne može previše uređivati dizajn stranice, te nam za to služi jezik zvan CSS. Pomoću CSS-a mogu se uređivati boje naslova, veličina i izgled fonta, boja pozadine, uređivanje listi, margina itd. Za stvaranje dinamičnog web sadržaja na web stranici ili aplikaciji koristi se JavaScript programski jezik. JavaScript upravlja multimedijom, ažuriranjem slika te se izvršava na neku akciju kao klik.

Nabrojani su najpoznatiji JavaScript razvojni okviri backenda i frontenda, kao Django, Ruby, AngularJS, Vue.js i ReactJS kao i JavaScript biblioteke kao što su jQuery i React library. U radu su analizirani popularni okviri AngularJS, Vue.js i React.js te njihove prednosti, nedostaci i značajke, kao i mogućnosti njihove integracije u postojeći sustav. AngularJS je samostalni frontend okvir s ugrađenim alatima i knjižnicama koje ne ometaju veličinu ili brzinu aplikacije. Vue.js je okvir koji pruža napredne web alate za razvoj modernističkih aplikacija na jednoj stranici i frontend web aplikacija. React.js je JavaScript biblioteka otvorenog koga za mobilne i web aplikacije.

U sklopu rada prikazane su sličnosti i razlike u korištenju okvira koji su analizirani u radu. AngularJS najbolji je izbor za dinamične projekte koji koriste većinu značajki razvojnih okvira, dok Vue.js je bolji za manje projekte i druge aplikacije koje preferiraju veću brzinu nego veću fleksibilnost, te React.js koji je brz i skalabilan za stvaranje velikih web aplikacija bez ponovnog učitavanja stranice. Statistički prikazano React.js vodeći je po svim kategorijama kao što su preuzimanje, korištenja kategorija web stranica, GitHub zvjezdica, itd. AngularJS slijedi React.js po istim navedenim kategorijama, dok Vue.js ipak ima manju korištenost osim u preuzimanju okvira i broju zvjezdica.

5. Literatura

1) AngularJS vs Vue.js (2021)

   Dostupno na : https://www.similartech.com/compare/angular-js-vs-vuejs (05.09.2022)

2) Angular Vs React Vs Vue: The Main Differences And Use Cases (2022)

   Dostupno na : https://incora.software/insights/react-vs-angular-vs-vue-the-main-differences-and-use-cases/55 (05.09.2022)

3) Boehrm, A. and Ruvalcaba, Z. : Murach's HTML and CSS. 5th ed. Fresno: Mike Murach & Associates Inc, 2022.

4) Delamater, M. and Ruvalcaba, Z.: Murach's JavaScript and jQuery. 4th ed. Fresno: Mike Murach & Associates Inc, 2020.

5) Djirdeh, H, Murray, N. and Lerner, A : Fullstack Vue: The Complete Guide to Vue.js. 1.st ed. CreateSpace Independent Publishing Platform, 2018.

6) Eisenman, B. : Learning React Native: Building Native Mobile Apps with JavaScript. 2nd ed. O'Reilly Media, 2017.

7) Google Trends (2022)

   Dostupno na: https://trends.google.com/trends/explore?cat=31&date=today%2012-m,today%2012-m,today%2012-m&geo=,,&q=Vue%20jobs,React%20jobs,Angular%20jobs

8) Harris, A. : HTML5 and CSS3 All-in-One For Dummies. 3rd ed. New York: John Wiley & Sons Inc, 2014.

9) Holzschlag, M., Skok u HTML i CSS. Beograd: Kompjuterska biblioteka, 2006.

10) Josh Hill, James A.Brannen : HTML5 and CSS3, 2011.

    https://www.knjizara.com/pdf/133144.pdf

11) Karpov. V and Netto, D : Professional AngularJS. 1st ed. Wrox, 2015.

12) Macrae, C. : Vue.js: Up and Running: Building Accessible and Performant Web Apps. 1st ed. Sebastopol : O'Reilly Media, Inc, 2018.

13) McGrath, M. : HTML, CSS and JavaScript in easy steps. Southam: In Easy Steps Limited, 2020.

14) Metzgar, D. : Exploring .NET Core with Microservices, ASP.NET Core, and Entity Framework Core. Manning, 2017

15) Meyer, E. A. : CSS Pocket Reference. 5th ed. Sebastopol : O'Reilly Media, Inc, 2018.

16) Piligrim, M., HTML 5: Spreman za upotrebu. Zagreb: Dobar plan, 2010.

17) Robbins, J. : Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics. 5th ed. Sebastopol : O'Reilly Media, Inc, 2018.

18) Seshadri, S and Green, B : AngularJS. 1st ed. Sebastopol : O'Reilly Media, Inc, 2013.

19) Star History (2022)

Dostupno na : https://star history.com/#facebook/react&vuejs/vue&angular/angular&Date

20) Stefanov, S. : React: Up & Running: Building Web Applications. 2nd ed. Sebastopol : O'Reilly Media, Inc, 2021

21) Vue vs React vs Angular: What Framework Would You Choose? (2020)

Dostupno na : https://medium.com/swlh/vue-vs-react-vs-angular-what-framework-would-you-choose-5d77a3680b0d (07.09.2022)

Razvoj klijentskog dijela web aplikacije

Sažetak

U radu su prikazane komponente aplikacije rađene u klijent-server arhitekturi, a poseban naglasak stavljen je na razvoj klijentskog dijela web aplikacije. Iako se klijentski dio temelji na tehnologijama poput HTML-a, CSS-a i JavaScripta, razvojni inženjeri vrlo često koriste okvire koji pružaju moderan pristup dizajnu i funkcionalnosti, te olakšavaju prikaz podataka dohvaćenih sa servera. U radu su analizirani popularni okviri, njihove prednosti, nedostaci i značajke, kao i mogućnosti njihove integracije u postojeći sustav. U sklopu rada su prikazane sličnosti i razlike u korištenju i sintaksi najpoznatijih razvojnih okvira, poput AngularJS, Vue.js i React.js.

Ključne riječi: HTML, CSS, JavaScript, Framework, AngularJS, Vue.js, React.js

Developing the web application frontend

Summary

The paper presents the components of the application made in the client-server architecture, and special emphasis will be placed on the development of the client part of the web application. Although the client part is based on technologies such as HTML, CSS and JavaScript, developers often use frameworks that provide modern access to design and functionality, and facilitate the display of data retrieved from the server. The paper presents and analyzes popular frameworks, their advantages, disadvantages and features, as well as the possibilities of their integration into the existing system. Also, the paper shows the similarities and differences in the usage and syntax of the most famous frameworks, such as AngularJS, Vue.js and React.js.

Key words: HTML, CSS, JavaScript, Framework, AngularJS, Vue.js, React.js