Osnove razvoja web sučelja i izrada chat aplikacije

Huljak, Karlo

SVEUČILIŠTE U ZAGREBU
FILOZOFSKI FAKULTET
ODSJEK ZA INFORMACIJSKE I KOMUNIKACIJSKE ZNANOSTI
Ak. god. 2022./2023.

Karlo Huljak

Osnove razvoja web sučelja i izrade chat aplikacije

Završni rad

Mentor: izv. prof. dr.sc. Vedran Juričić

Zagreb, rujan 2023.

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.

Sadržaj

1. UVOD

Većina korisnika interneta ili internetskih aplikacija susrela se s nekim oblikom web sučelja. Ta web sučelja, u moderno doba interneta, većinom su interaktivna. To znači da nisu statična, već reagiraju na radnje koje korisnik vrši. Obično, za izradu takvih sučelja koriste se tehnologije kao što su HTML, CSS i Javascript, a pristup izrađenom sučelju omogućuju web preglednici. Da bi se olakšalo korištenje ovih tehnologija, ali u isto vrijeme i poboljšala kvaliteta web sučelja, koriste se razvojne okoline. Jedna od najpopularnijih razvojnih okolina je ReactJS, koja se bazira na već spomenutim tehnologijama, HTML-u, CSS-u i na programskom jeziku Javascriptu. HTML se koristi kako bi se na web sučelje stavio sadržaj. Dakle, HTML dokument će prikazati sadržaj web sučelja bez ikakvog stilskog oblikovanja ili funkcionalnosti. Da bi taj sadržaj bio stilski oblikovan, koristi se CSS. CSS daje sadržaju različite osobine kao što su pozicija na ekranu ili boja. Zatim, da bi web sučelja reagirala na neku akciju korisnika, koristi se Javascript. Ovaj rad objašnjava osnovne koncepte ovih tehnologija, ali i daje primjer korištenja istih u obliku stvaranja chat aplikacije. No, prije toga, valja objasniti na koji način web preglednici prikazuju web sučelje korisniku.

2. Osnovne tehnologije za razvoj

Korisnici najčešće imaju pristup internetu preko web preglednika. Web preglednici čitaju upute iz HTML dokumenta i prikazuju sadržaj na web stranici. HTML oznake se nalaze u tekstualnoj datoteci koja sadrži upute za preglednik. Ključna uloga u ovoj web slagalici pripada web pregledniku, jer čita upute napisane u HTML-u i koristi ih za prikazivanje sadržaja web stranice na ekranu korisnika. (Tittel & Burmeister, 2008, 13) Obično postoje dva izvora HTML datoteka iz kojih računalo može dobiti upute o sadržaju web stranice: web serveri ili lokalno računalo. Dakle, prije nego što web stranice postanu dostupne na internetu, mogu se prikazati i lokalno, na osobnom računalu. Zatim, ako je web stranica namijenjena drugim računalima, oni će je dobiti preko web servera. (Tittel & Burmeister, 2008, p. 14)

Serveri su računala koja služe za pohranjivanje i prijenos informacija. Server pruža uslugu, odnosno daje web pregledniku informacije, a web preglednik od tih informacija gradi web stranicu:

Web poslužitelj je računalo koje:

1. Se povezuje s internetom
2. Pokreće web poslužiteljski softver
3. Odgovara na zahtjeve web preglednika za web stranicama

Gotovo svako računalo može biti iskorišteno kao web poslužitelj, uključujući i kućna računala. Međutim, web poslužitelji su obično računala koja su namijenjena isključivo toj svrsi. Za objavu vlastitih web stranica nije potrebno biti stručnjak za internet ili računarstvo, ali treba pronaći web poslužitelja koji će poslužiti te stranice. (Tittel & Burmeister, 2008, p. 14)

Jedna od najvažnijih i najvrjednijih značajki interneta je mogućnost da uspostavi vezu između različitih web stranica. Te veze nazivaju se poveznicama. Specifične HTML upute omogućuju tekstu da upućuje (poveže) s drugim sadržajem. Takvi pokazatelji su poznati kao hiperlinkovi. Hiperlinkovi predstavljaju vezivo koje održava World Wide Web povezanim. Obično se u web preglednicima hiperlinkovi prikazuju plavom bojom i podcrtani su. Klikom na njih, korisnici se preusmjeravaju na drugo mjesto. World Wide Web nosi svoje ime doslovno, jer predstavlja mrežu stranica smještenih na web poslužiteljima diljem svijeta, međusobno povezanih na milijune načina. Te veze se ostvaruju putem hiperlinkova koji povezuju jednu stranicu s drugom. Bez takvih veza, web bi bio samo skup samostalnih stranica (Tittel & Burmeister, 2008, p. 10)

Dakle, gotovo sve web stranice su povezane s drugim web stranicama preko poveznica. No, to omogućuje još jedno važno obilježje interneta. Svaka web stranica, ponekad čak i datoteka, ima svoju vlastitu, jedinstvenu web adresu. Ta jedinstvena web adresa naziva se URL. URL se može upisati u tražilicu web preglednika kako bi se našla željena stranica ili datoteka. Web se sastoji od milijuna resursa, svaki od njih povezan sa svojom točnom lokacijom. Ta precizna adresa, poznata kao Uniform Resource Locator (URL), ključna je za uspješno povezivanje s tim resursima. Bez ispravnog URL-a, nemoguće je koristiti adresnu traku web preglednika za izravan pristup web stranici. URL-ovi predstavljaju standardni sustav adresiranja za resurse na internetu. Svaki resurs, bilo da se radi o web stranici, web sjedištu ili pojedinačnoj datoteci, ima svoj jedinstveni URL. URL-ovi funkcioniraju slično kao i poštanska adresa. (Tittel & Burmeister ,15)

Svaki URL sastoji se od četiri elementa, a to su protokol, domena, put i ime datoteke:

1. Protokol: Određuje protokol koji web preglednik koristi za zahtjev za datoteku. Protokol za web stranicu je http:// (uobičajeni početak većine URL-ova).

2. Domena: Pokazuje na općenitu web stranicu (kao što je www.sun.com) gdje se datoteka nalazi. Domena može sadržavati nekoliko datoteka (kao što je osobna web stranica) ili milijune datoteka (kao što je korporativna stranica, poput www.sun.com).

3. Putanja: Imenuje niz mapa kroz koje morate navigirati kako biste došli do određene datoteke. Na primjer, da biste došli do datoteke u mapi evangcentral koja se nalazi u mapi developers, koristite putanju /developers/evangcentral/.

4. Ime datoteke: Označava koju datoteku u putanji direktorija web preglednik pristupa.(Tittel & Burmeister, 2008, p. 15)

2.1. Uvod u HTML

HTML se koristi na većini internetskih stranica, a služi za prikaz informacija i sadržaja. Na web stranicama se može pronaći raznoliki sadržaj, uključujući tekst, grafike, obrasce, audio i video datoteke, te interaktivne igre. Svaka web stranica je jedinstvena, ali većina dijeli zajednički element: Hypertext Markup Language (HTML). (Tittel & Burmeister, 2008, p. 10)

Dakle, sav tekst i grafika, npr. slike, su dio HTML-a. HTML ne uključuje stilsko uređivanje web sučelja niti davanje funkcionalnosti tom sučelju. Stoga, HTML dokumente se može smatrati jednostavnim tekstualnim dokumentima. To je jedan od razloga zašto se i toliko koristi, računalima je ‘lakše’ komunicirati jednostavnim tekstualnim dokumentima. Bez obzira

na sadržaj koji se nalazi na web stranici, svaka web stranica je temeljena na HTML-u (ili ekvivalentnom jeziku). HTML predstavlja vezivni materijal koji održava web stranicu integriranom; slično kako cigle služe za građenje zida. HTML datoteke koje oblikuju web stranice su čisto tekstualni dokumenti. To je razlog zbog kojeg web funkcionira tako besprijekorno. Tekst je univerzalni jezik računala, pa bilo koja tekstualna datoteka stvorena na Windows računalu, uključujući HTML datoteke, podjednako dobro funkcionira na bilo kojem drugom operativnom sustavu.“ (Tittel & Burmeister, 2008, p. 10)

S obzirom na to da se HTML koristi samo za sadržajni dio web stranica, sintaksa nije pretjerano kompleksna u usporedbi s nekim drugim programskim alatima ili jezicima. HTML sintaksa koristi elemente, atribute i entitete, od kojih su najvažniji elementi i atributi:

1. Elementi: Identificiraju različite dijelove HTML stranice korištenjem oznaka.
2. Atributi: Informacije o instanci elementa.
3. Entiteti: Tekstualni znakovi koji nisu ASCII znakovi, kao što su znakovi za autorska prava (©) i akcentirana slova. (Tittel & Burmeister , 19)

Elementi

Elementi su, uz atribute, najvažniji dio HTML sintakse. Označeni su 'tagovima', a primjer elementa izgleda ovako: <tag></tag>. Postoje dvije glavne vrste elemenata:

1. Elementi s sadržajem koji se sastoji od para oznaka i bilo kojeg sadržaja koji se nalazi između početne i završne oznake u paru.

2. Elementi koji umetnu nešto na stranicu koristeći samo jednu oznaku.

Unutar početnog i završnog 'taga' elementa nalazi se sadržaj kao što su paragrafi, zaglavlja, tablice i liste. Sadržaj - kao što su odlomci, naslovi, tablice i liste - uvijek koristi par oznaka:

1. Početna oznaka (<tag>) govori pregledniku, 'Element počinje ovdje.'
2. Završna oznaka (</tag>) govori pregledniku, 'Element završava ovdje.' (Tittel & Burmeister, 2008, p. 20)

No, postoje i elementi koji zahtijevaju samo jedan 'tag': Elementi koji umetnu nešto na stranicu nazivaju se praznim elementima (jer ne sadrže nikakav sadržaj) i koriste samo jednu oznaku, poput ove: <tag />. Nadalje, jedan od bitnih koncepata u HTML sintaksi jest ugnježđivanje

elemenata. Mnoge strukture web stranica uključuju elemente koji su ugniježđeni jedan unutar drugoga. Ti ugniježđeni elementi mogu se zamisliti kao kutije koje se uredno stavljaju unutar drugih kutija. Na primjer, popis s oznakama koristi dvije vrste elemenata: <ul> element označava da se radi o popisu bez reda (s oznakama), dok <li> elementi označavaju svaku stavku na tom popisu.(Tittel & Burmeister, 2008, p. 21) Jedan od najboljih primjera ugnježđivanja elemenata su liste. predstavlja početak neorganizirane liste, unutar tagova <li> i nalazi se jedan elemenata te liste, a označava kraj liste, kao što je prikazano u programskom isječku 1.

```
 <ul>
<li>Barbera</li>
<li>Brunello</li>
</ul>
```

Programski isječak 1. Neorganizirana lista

Atributi

Atributi se mogu smatrati dodatnim opisom sadržaja nekog elementa. Atributi omogućuju raznolikost u načinu na koji element opisuje sadržaj ili radi. Atributi vam omogućuju da koristite elemente na različite načine ovisno o okolnostima.(Tittel & Burmeister, 2008, p. 21)

Na primjer, element <img /> koristi atribut 'src' kako bi odredio lokaciju slike koja se želi uključiti na određenoj poziciji na svojoj stranici. Primjer se može vidjeti u programskom isječku 2.

```
<img src=”red_grapes.jpg” width=”75” height=”100” alt=”Red
Grapes” align=”middle” hspace=”5” />
```

Programski isječak 2. Element koji sadrži više atributa

Dakle, <img/> označava element, odnosno sliku, a sve ostalu unutar tog 'taga' su atributi koji web pregledniku daju informacije o toj slici, kao što su izvor slike, pozicija na ekranu i opis slike .U ovom dijelu HTML-a, sam element <img /> općenito obavještava preglednik da želite uključiti sliku; atribut “src” pruža pojedinosti o slici koju želite uključiti - u ovom slučaju, red_grapes.jpg. Drugi atributi (kao što su širina, visina, poravnanje i hspace) pružaju informacije o tome kako prikazati sliku, dok atribut alt pruža tekstualnu alternativu slici koju tekstualni preglednik može prikazati. (Tittel & Burmeister, 2008, p. 22)

Ako postoje dva 'taga', atributi se uključuju prije završetka početnog 'taga', kao što je prikazano u programskom isječku 3.

```
<tag attribute=”value” attribute=”value”>
```

Programski isječak 3. Sintaksa sještanja atributa unutar elementa

2.2. Uvod u CSS

Nakon dodavanja sadržaja koristeći HTML, taj sadržaj treba stilski oblikovati kako bi web sučelje bilo estetski zadovoljavajuće. Upravo se za taj posao koristi CSS. CSS se koristi za definiranje stilova za vaše web stranice, uključujući dizajn, raspored i varijacije prikaza na različitim uređajima i veličinama zaslona.(*CSS Syntax)* CSS sintaksa referira HTML sinatksu i zadaje atribute određenim HTML elementima, vidljivo u programskom isječku 4.

```
p {
  color: red;
  text-align: center;
}
```

Programski isječak 4. Referiranje HTML elementa pomoću CSS sintakse

"p" je selektor u CSS-u (usmjerava se prema HTML elementu koji želite stilizirati: <p>). "color" je svojstvo, a "red" je vrijednost svojstva. "text-align" je svojstvo, a "center" je vrijednost svojstva. (*CSS Syntax)*

CSS koristi takozvane 'selektore' kako bi se odredilo na koje se elemente odnosi određena CSS sintaksa. U prošlom primjeru, CSS sintaksa se odnosi na sve <p> elemente u HTML dokumentu. No, ako se nekom elementu u HTML dokumentu doda atribut 'id', CSS može se primijeniti samo na taj element. U programskom isječku 5 CSS pravilo će se primijeniti na HTML element s id="para1".

```
#para1 {
text-align: center;
color: red;
}
```

Programski isječak 5. Referiranje na HTML element pomuću CSS sintakse koristeći atribut ‘id’

Isto tako, ako se elementu doda atribut 'class', CSS može referirati samo na taj određeni element, odnosno sve elemente koji dijele istu vrijednost za atribut 'class', kao što je prikazano u programskom isječku 6.

```
.center {
text-align: center;
color: red;
}
```

Programski isječak 6. Referiranje na HTML element pomuću CSS sintakse koristeći atribut 'class'

Dakle, CSS se koristi za stilsko obilježavanja sadržaja i definiranje položaja sadržaja koji se nalazi u HTML dokumentu. Do tog sadržaja dolazi se pomoću HTML 'tagova' ili pomoću atributa 'id' i 'class'.

2.3. Uvod u Javascript

Javascript je programski jezik kojeg web preglednici koriste kako bi web sučelja dobila na funkcionalnosti. S obzirom na to da se Javascript koristi kompilacijom samo u trenutku izvršavanja (JIT), kod koji se preuzima preglednikom prilikom posjeta web stranici je izvorni kod. To znači da je moguće da bilo tko tko pristupa bilo kojoj web stranici može čitati i preuzeti kod (uključujući HTML, CSS i Javascript) koji čini funkcionalnost korisničkog sučelja web stranice. (Minnick, 2023, p. 15)

Da bi se otkrila uloga Javascripta u svijetu weba, treba istražiti kako su računala komunicirala putem interneta prije korištenja ovog programskog jezika. Internet je mreža sastavljena od milijuna računala koja mogu međusobno komunicirati putem skupa protokola poznatih kao TCP/IP. Skraćenica TCP/IP predstavlja Transmission Control Protocol / Internet Protocol i definira kako se podaci prenose putem interneta i kako bilo koje računalo može pronaći druga računala povezana na mrežu. Sve računalo povezano s internetom može izvoditi aplikacije koje šalju podatke preko interneta. Jedna od takvih aplikacija je HTTP poslužitelj. Zadatak HTTP poslužitelja je koristiti HTTP protokol (ili, danas češće, njegovu enkriptiranu verziju, HTTPS) kako bi odgovarao na zahtjeve za web stranicama i drugim datotekama potrebnim za izgradnju web stranica. Zahtjevi prema web poslužiteljima često (iako ne nužno) dolaze od programa poznatih kao web preglednici. (Minnick, 2023, p. 16) Dakle, kada web preglednik zatraži

određenu web stranicu, koristi se HTTPS protokol kako bi se od servera zatražila ta stranica. U web pregledniku koji ne koristi Javascript, preglednik šalje zahtjev prema web poslužitelju putem HTTP protokola. Svaki put kada korisnik klikne na vezu ili unese adresu u traku za adresu preglednika, obavještava svoju klijentsku aplikaciju (web preglednik) da treba poslati zahtjev serveru. Na primjer, ako korisnik zatraži web stranicu na adresi http://www.example.com/about.html, obavještava preglednik da treba zatražiti dokument po imenu about.html sa IP adresom koja odgovara domeni www.example.com. (Minnick, 2023, p. 17) Međutim, bez korištenja Javascripta, sve što web preglednik dobije od servera nakon što zatraži određenu web stranicu je HTML datoteka. Pojavom Javascripta, preglednici su stekli veće sposobnosti za manipulaciju podacima primljenim s poslužitelja. Sada, umjesto jednostavnog prikazivanja HTML datoteke koju je dostavio poslužitelj, preglednici mogu preuzeti skripte i podatke s poslužitelja koji omogućuju obradu podataka, njihovo reorganiziranje i mnogo više. Razmotrite web aplikacije poput Gmaila, na primjer. Ako ostavite otvoren prozor preglednika s Gmailom, nove poruke se automatski preuzimaju. Možete sortirati i filtrirati svoje poruke, pisati nove poruke i izvoditi različite zadatke. (Minnick,2023, p. 17)

Dakle, Javascript je danas gotovo neizostavan dio web stranica. Bez tog programskog jezika, web stranice bile bi statične, ne bi se automatski ažurirale niti bi postojale mnoge značajke modernih web stranica. Za pisanje Javascript sintakse često se koriste razni alati, a Visual Studio Code jedan je od najpopularnijih. Ključni alat za svakog programera je uređivač koda, gdje se piše izvorni kod koji čini program. Visual Studio Code, ili VS Code, predstavlja općeniti uređivač otvorenog koda namijenjen web razvoju. Iako postoje brojne druge opcije uređivača koda, većina Javascript programera trenutno koristi VS Code. (Minnick, 2023, p. 22) Visual Studio Code koristio se i izradi chat aplikacije koja je opisana u ovom radu.

Osnovna jedinica sintakse Javascripta naziva se izjavom. Baš kao što je rečenica u ljudskom jeziku temeljna građevna jedinica svakog teksta, Javascript ima sličnu temeljnu građevnu jedinicu, koja se naziva izjava (statement). (Minnick, 2023, p. 30)

Izjave se sastoje od vrijednosti, operatora, ključnih riječi i ekspresija. U programskom isječku 7 nalazi se primjer jednostave Javascript sintakse.

```
let normalName = 'Chris';
let JavascriptName = normalName + 'Script';
console.log('Your Javascript Name is ' + JavascriptName);
```

Programski isječak 7. Stvaranje i rad s varijablama pomoću Javascript sintakse

Kao što engleski jezik sastoji od dijelova govora - kao što su imenice, glagoli i prilozi, Javascript izjave sastoje se od nekoliko dijelova, kako je opisano u ovom popisu:

- Vrijednosti: 'Chris' je vrijednost. Izrazi: Izrazi su jedinice koda koje se vrednuju i postaju vrijednosti. Sljedeći je izraz: normalName + 'Script'. Kada normalName ima vrijednost 'Chris', ovaj izraz se vrednuje kao 'ChrisScript'.
- Operatori: Operatori nešto rade s vrijednostima. Operator + (poznat kao operator konkatenacije) spaja vrijednost normalName i doslovnu vrijednost 'Script'.
- Ključne riječi: Ključne riječi su dijelovi Javascripta koji imaju posebno značenje i potiču Javascript da nešto radi. Riječ 'let' je ključna riječ koja govori Javascriptu da pohrani vrijednost s desne strane operatora = (operator dodjele) koristeći ime s lijeve strane operatora“ =”. (Minnick, 2023, p. 30)

Kod pisanja Javascript sintakse, valja biti oprezan u vezi velikih i malih slova. Naime, Javascript razlikuje veliko i malo slovo te, ako se zamjene, Javascript neće 'razumjeti' namjenu koda. Ako se promijeni veličina slova u jednom znaku u Javascript ključnoj riječi ili imenu varijable, Javascript ne razumije što želite reći. (Minnick, 2023, p. 31)

Nizovi

Za pohranjivanje informacija u jednu cjelinu, koriste se nizovi: Često se javlja potreba za uređenom kolekcijom, gdje imamo 1., 2., 3. element i tako dalje. Na primjer, to nam je potrebno kako bismo pohranili popis nečega: korisnika, dobara, HTML elemenata itd.(Arrays)

Postoji posebna podatkovna struktura nazvana polje (Array) koja služi za pohranu uređenih kolekcija. Primjer niza bio bi:

```
const boje = ['plava', 'crvena', 'zelena'];
```

Nadalje, postoji nekoliko osnovnih informacija o nizovima u Javascriptu koje se moraju znati kako bi se nizovi mogli pravilno koristiti. Ovaj niz ima tri elementa, pa se kaže da ima duljinu od 3. Elementi niza numeriraju se počevši od 0. To se naziva numeriranje počevši od nule (zero-based numbering). Prototip za niz je objekt Array, što znači da svaki niz koji stvorite ima

pristup određenim svojstvima i metodama definiranim u objektu Array. Sve te atribute i metode možete vidjeti klikom na strelicu u konzoli da biste proširili objekt Prototip (Prototype).(Minnick, 2023, p. 107) Dakle, postoji objekt Array koji ima definirana svojstva, i svaki niz je zapravo jedna instanca tog objekta. No, da bi se objekt Array mogao instancirati, potrebno je koristiti konstruktor. Konstrukcijska funkcija je funkcija koja stvara i inicijalizira objekt. U slučaju konstrukcijske funkcije Array(), stvara se novi objekt niza. Za korištenje konstrukcijske funkcije, operator new se koristi zajedno s imenom konstrukcijske funkcije, nakon čega slijede otvorene i zatvorene zagrade: new Array(); (Minnick, 2023, p. 108)

Nadalje, u prethodnom nizu, `const boje = ['plava', 'crvena', 'zelena'];`, 'plava' ima indeks 0, 'crvena' ima indeks 1 i 'zelena' ima indeks 2. Dakle, ako se želi pretražiti niz, i pronaći element koji ima indeks 1 unutar tog niza, koristit će se sintaksa: boje[1]. No, postoji velik broj različitih metoda koje se mogu koristiti kad su u pitanju nizovi, a sve se mogu naći u tablici 1.

concat(): Stvara novi niz koji se sastoji od trenutnog niza spojenog s drugim nizovima i/ili vrijednostima.
every(): Vraća true ako svaki element u danoj nizu zadovoljava zadanu testnu funkciju. filter(): Stvara novi niz s svim elementima trenutnog niza koji zadovoljavaju zadatu funkciju za testiranje.
forEach(): Izvršava funkciju za svaki element u nizu. includes(): Utvrđuje da li niz sadrži određenu vrijednost i vraća true ili false.
indexOf(): Pronalazi prvo pojavljivanje određene vrijednosti u nizu i vraća indeks tog elementa; vraća -1 ako vrijednost nije pronađena. join(): Spaja sve elemente niza u jedan string
lastIndexOf(): Pronalazi posljednje pojavljivanje određene vrijednosti u nizu i vraća indeks tog elementa; vraća -1 ako vrijednost nije pronađena. map(): Stvara novi niz s rezultatom primijenjenim određenom funkcijom na svaki element u nizu. pop(): Uklanja posljednji element iz niza.
push(): Dodaje nove stavke na kraj niza.
reduce(): Smanjuje vrijednosti u nizu u jednu vrijednost primjenom funkcije (s lijeva na desno).

reverse(): Okreće redoslijed elemenata u nizu.
shift(): Uklanja prvi element iz niza i vraća taj element, što rezultira promjenom duljine niza.
slice(): Odabire dio niza i vraća ga kao novi niz.
some(): Vraća true ako jedan ili više elemenata zadovoljava zadanu testnu funkciju.
sort(): Stvara novi niz sortiranjem elemenata u nizu.
splice(): Vraća novi niz sastavljen od elemenata koji su dodani ili uklonjeni iz danog niza.
toString(): Pretvara niz u string.
unshift(): Vraća novi niz s novom duljinom dodavanjem jednog ili više elemenata na početak niza.

Tablica 1. Popis metoda u Javascript sintaksi

Objekti

Još jedan od načina pohranjivanja informacija u Javascriptu su Javascript objekti. Objekti su ponovno upotrebljive komponente koje sadrže podatke i funkcionalnost. Objekti u stvarnom svijetu imaju karakteristike koje ih definiraju. Na primjer, olovka ima duljinu, promjer, boju i druge karakteristike koje je opisuju. Olovka također ima stvari koje možete s njom raditi, poput pisanja, brisanja, šiljenja i lomljenja. programskom isječku 8 prikazuje kako bi izgledalo kada bi se pojam olovke postavio kao Javascript objekt. (Minnick, 2023, p. 125)

```
const pencil = {
length: „7.5 inches“,
shape: „hexagonal“,
 diameter: „1/4 inch“,
 write: function(){/*do writing*/},
 erase: function(){/*do erasing*/},
 sharpen: function(){/*do sharpening*/},
}
```

Programski isječak 8. Javascript objekt

Objekti se obično sastoje od svojstva i metoda. U Javascriptu, kako bi se opisali podaci i funkcionalnosti sadržane unutar objekta, koristi se pojam "svojstva" (properties). Ako svojstvo

ima vrijednost u obliku funkcije, naziva se "metodom" (method). Javascript objekti mogu se koristiti kako bi se opisali fizički objekti, ali također ih je moguće koristiti kako bi se opisale apstraktne ideje.(Minnick, 2023, p. 126) Metode koje su definirane unutar objekta su zapravo funkcije koje se mogu definirati i zatim koristiti. Nakon što je stvoren objekt olovke, informacije o njemu se mogu saznati, ali i koristiti njegove funkcije pomoću tzv. "točkaste notacije" (dot notation). Na primjer, kako biste oštrila olovka, može se pozvati metoda sharpen() olovke pomoću sljedećeg koda:

```
pencil.sharpen();
```

Ovdje, pencil je objekt olovke, a sharpen je metoda objekta koja izvršava radnju oštrenja olovke. Korištenjem točkaste notacije, može se pristupiti i pozivati različite funkcionalnosti objekta. Postoje četiri načina kreiranja objekta u Javascriptu: „Koristeći notaciju doslovnog objekta, koristeći ključnu riječ new, koristeći Object.create() i definiranjem klase.(Minnick, 2023, p. 127) Programski isječak 9 daje primjer kreiranja objekta koristeći notaciju doslovnog objekta. Primjer kreiranja objekta pomoću ključne riječi 'new' nalazi se u programskom isječku 10. Zatim, programski isječak 11 daje primjer stvaranja objekta pomoću klase, a 12 primjer kreiranja objekta pomoću Object.create().

```
 „const person = {};
 person.hair = 'black';
person.hands = 2;
 person.fullName = {firstName:'Lamont',lastName:'Rudnick'};
```

Programski isječak 9. Notacija doslovnog objekta

```
function Cat(name, type){
 this.name = name;
 this.type = type;
 }
const ourCat = new Cat('Murray', 'domestic short hair');
```

Programski isječak 10. Stvaranje objekta pomoću ključne riječi 'new'

```
class Pet { constructor(name,type)
 this.name = name;
this.type = type; }}
```

```
const ourDog = new Pet('Chauncey','AmStaff');
```

Programski isječak 11. Stvaranje objekta pomoću klase

```
const computer = {memory:'16GB',HD:'8TB'}
const laptop = Object.create(computer);
```

Programski isječak 12. Stvaranje objekta pomoću Object.create()

Nakon što je objekt stvoren, moguće je pristupiti, mijenjati i dodavati nova svojstva putem jedne od dviju metoda: notacije sa točkom ili notacije s uglatim zagradama. U notaciji sa točkom (dot notacija), ime svojstva slijedi nakon točke, a zatim slijedi ime svojstva ili metode. Prednost notacije sa točkom leži u tome što olakšava pristup svojstvima i metodama ugniježđenih objekata.(Minnick, 2023, p. 130)

Slijedi primjer modificiranja objekta pomoću notacije sa točkom u programskom isječku 13.

```
const myLocation = { city: {
 id: 2643743,
 name: 'London',
 coord: { lon: -0.1258, lat: 51.5085, },
 country: 'GB',
 population: 9820000,
 timezone: 3600, }, };
```

Programski isječak 13. Modificiranje objekta pomoću notacije sa točkom

Drugi način korištenja objekta je notacija s uglatim zagradama. Notacija s uglatim zagradama koristi upravo uglate zagrade nakon imena objekta kako bi se dobile i postavile vrijednosti svojstava. Ime svojstva između tih uglatih zagrada je string, što znači da može biti doslovni string ili ime varijable koja sadrži string. (Minnick, 2023, p. 131)

Primjer korištenja notacije s uglatim zagradama bio bi '*myLocation['city']*'. Kod objašnjavanja koncepta Javascript objekta, korisno je znati i ulogu prototipa. Svaki objekt u Javascriptu nasljeđuje svojstva i metode od svojeg prototipa. Na primjer, kada stvorite niz, on nasljeđuje svojstva i metode iz Array.prototype. Sam Array potom nasljeđuje svojstva iz Object.prototype. Kada stvorite novi objekt putem doslovne notacije, taj objekt nasljeđuje

svojstva iz Object.prototype. Kada koristite konstrukcijsku funkciju ili razred za stvaranje objekta, taj razred ili konstruktor postaje prototip tog objekta..(Minnick, 2023, p. 134)

Funkcije

Javascript funkcije su specifična vrsta objekta koji omogućuju funkcionalnost ovog programskog jezika. Funkcije su objekti koji omogućuju programima da izvršavaju određene zadatke i igraju važnu ulogu u organizaciji koda. Slično tome kako objekti organiziraju podatke u ponovno iskoristive kontejnere, funkcije organiziraju izjave u ponovno iskoristivu funkcionalnost. Funkcija predstavlja grupu izjava koja izvršava specifični zadatak, tj. funkcija je poput manjeg programa unutar vašeg programa. Obično, funkcije primaju ulazne podatke i zatim vraćaju izlazne podatke na temelju tih ulaza.(Minnick, 2023, p. 139)

Dakle, korisnik može samostalno definirati takozvane 'top-level' funkcije, no Javascript sadrži i neke predefinirane funkcije koje se mogu koristiti, a prikazane su u tablici 15.

Boolean(): Pretvara vrijednost koja nije boolean u boolean.
Number(): Pretvara vrijednost koja nije broj u broj.
String(): Pretvara vrijednost koja nije string u string.
eval(): Izvršava Javascript kod proslijeđen kao niz znakova.
uneval(): Stvara string iz izvornog koda proslijeđenog kao niz znakova.
isFinite(): Utvrđuje je li vrijednost koja joj je proslijeđena konačan broj.
isNaN(): Utvrđuje je li vrijednost koja joj je proslijeđena NaN (nije broj).
parseFloat(): Pretvara string u decimalni broj s pomičnim zarezom.
parseInt(): Pretvara string u cjelobrojni broj.
decodeURI(): Dekodira string koji je prethodno kodiran pomoću encodeURI().
decodeURIComponent(): Dekodira string koji je prethodno kodiran pomoću encodeURIComponent().
encodeURI(): Zamjenjuje određene znakove (npr. razmake, navodnike i kosu crtu) u stringu s nizovima za izbjegavanje kako bi se stvorio valjani Uniform Resource Identifier (URI), koji su adrese koje se koriste za lociranje web stranica i drugih resursa na webu.

encodeURIComponent(): Radi istu stvar kao i encodeURI() (vidi prethodni unos) ali kodira cijeli string, dok encodeURI() ignorira prefiks protokola (npr. http://) i domensko ime.“(Minnick, 2023, p. 141)

Tablica 2. Predefinirane Javascript funkcije

U primjeru koji slijedi, odnosno u programskom isječku 14, funkcija uzima argument koji se zatim koristi za odrađivanje neke radnje.

```
const favoriteFood = 'tacos';
makeDinner(favoriteFood);
function makeDinner(whatToMake) {
 console.log(`I see you want ${whatToMake}.`);
 whatToMake = 'salad';
 console.log(`I've decided to make ${whatToMake} instead.`);
}
```

Programski isječak 14. Javascript funkcija koja uzima argument

Dakle, prvo se postavi vrijednost varijable 'favoriteFood', a ta vrijednost je 'tacos'. Zatim, funkcija 'makeDinner' uzima vrijednost te varijable, odnosno 'tacos', i s njom vrši neku radnju. Kad su u pitanju argumenti funkcije, oni ne moraju biti samo neka vrijednost – na mjesto argumenta može doći i neka druga funkcija. Bilo koja vrijednost može biti proslijeđena funkciji kao argument, uključujući i druge funkcije. Kada se funkcija proslijedi drugoj funkciji kako bi je pozvala funkcija koja je proslijeđena, funkcija koja se prosljeđuje naziva se povratnom funkcijom (callback funkcijom). Funkcija koja se prosljeđuje povratnoj funkciji naziva se vanjskom funkcijom. (Minnick, 2023, p.146)

Primjer funkcije koja uzima neku drugu funkciju kao argument nalazi se u programskom isječku 15.

```
function greetInSpanish(name){
 return `Hola, ${name}`;
}
function getUserName(callback){
 let firstName = prompt('Enter your first name');
 return callback(firstName);
}
getUserName(greetInSpanish);
```

Programski isječak 15. Javascript funkcija uzima drugu funkciju kao argument

U ovom primjeru, funkcija 'getUserName' uzima funkciju 'greetInSpanish' kao argument. Osim ovog načina pisanja funkcija, postoje i takozvane 'streličaste' funkcije. Anonimne funkcije također se mogu napisati koristeći streličastu sintaksu. Streličasta sintaksa ne koristi ključnu riječ "function". Umjesto toga, koristi kombinaciju simbola koji izgledaju kao strelica, =>, između liste parametara i tijela funkcije.(Minnick, 2023, p.152) Primjerice, programski isječak 16 prikazuje na koji se način funkcija pretvara u streličastu funkciju.

```
const pickAMovie = function (choices) {
 Let    myPick    =    choices[Math.floor(Math.random()    *
choices.length)];
 return myPick;
};
const pickAMovie = (choices) => {
 let    myPick    =    choices[Math.floor(Math.random()    *
choices.length)];
 return myPick;
};
```

Programski isječak 16. Pretvaranje obične funkcije u streličastu funkciju

No, valja biti oprezan kod korištenja 'streličastih' funkcija, jer postoje dva ograničenja. Za razliku od funkcija stvorenih koristeći ključnu riječ "function", streličaste funkcije nemaju vlastiti "this". Umjesto toga, preuzimaju kontekst objekta u kojem su stvoreni, i njihova vrijednost "this" se ne mijenja kada se pozivaju u različitom kontekstu. Kao i kod običnih funkcija, možete proslijediti argumente streličastim funkcijama. Međutim, streličaste funkcije nemaju vlastiti objekt "arguments".(Minnick, 2023, p. 154)

Metode

JavaScript metode su akcije koje se mogu izvoditi nad objektima. JavaScript metoda je svojstvo koje sadrži definiciju funkcije, a prepoznatljive su jer koriste ključnu riječ *this*.(*JavaScript methods)*

Slijedi primjer metode, odnosno korištenja funkcije unutar objekta u programskom isječku 19.

```
const myCar = {
 speed: 0,
```

```
 drive(speedLimit) {
 this.speed = speedLimit;
 console.log(`Driving at ${this.speed}mph.`);
 },
};
```

Programski isječak 17. Javascript metoda unutar objekta

3. Razvojne okoline

Razvojne okoline temeljene na Javascript programskom jeziku drže ključnu ulogu u suvremenom razvoju front-end web aplikacija. Razvojne okoline su, u svojoj suštini, alati koji olakšavaju korištenje Javascripta, zajedno sa ostalim front-end tehnologija u svrhu ubrzavanja i olakšavanja procesa razvoja SPA aplikacija. SPA znači *'Single page application'*, odnosno, označava web aplikaciju koja učita samo jedan dokument, a zatim ažurira sadržaj tijela tog dokumenta.(*SPA (Single-page application*)) Dakle, ako se za primjer uzme chat aplikacija čiji je razvoj opisan u ovome radu, korisničko sučelje prikazuje samo jedan dokument te aplikacije, ali se sadržaj tog dokumenta mijenja ovisno o interakciji korisnika.

3.1. ReactJS

Jedna od najpoznatiji Javascript razvojnih okolina je ReactJS. ReactJS koristi HTML, CSS i Javascript za izgradnju i manipuliranje web sučeljem. Kada je ReactJS (poznat i kao React) bio potpuno nov, predstavljao je nov način izrade i ažuriranja korisničkih sučelja.(Minnick, 2023, p. 263) Primarni cilj React-a je smanjiti broj grešaka koje se pojavljuju prilikom izrade korisničkih sučelja (UI) od strane programera. To postiže kroz upotrebu komponenata - samostalnih, logičkih dijelova koda koji opisuju određeni dio korisničkog sučelja. Te komponente mogu se kombinirati kako bi se stvorilo potpuno korisničko sučelje, a React u pozadini odrađuje veći dio posla vezanog za renderiranje, što programeru omogućuje da se usredotoči na posao vezan uz dizajn korisničkog sučelja.(*Getting started with React*) React ne nameće stroge pravila kada su u pitanju konvencije za kodiranje ili organizacija datoteka. Ta fleksibilnost omogućava timovima da uspostave vlastite konvencije koje najbolje odgovaraju njihovim potrebama i da React uključe u svoje projekte onako kako smatraju da je najprikladnije. React može služiti kao alat za izradu pojedinačnog gumba, nekoliko elementa korisničkog sučelja ili cijelo korisničko sučelje. Iako se React može koristiti za manje komponente korisničkog sučelja, to se može zakomplicirati, stoga je React najjednostavniji za korištenje kad se pomoću njega izrađuje cijela aplikacija. (*Getting started with React*)

Postoji dvije vrste komponenata, tzv. *class* komponente i *function* komponente. U klasičnoj komponenti (class komponenti), nužno je uključiti izjavu "extends React.Component". Ova izjava stvara naslijeđivanje od React.Component-a i omogućuje komponenti pristup funkcijama React.Component-a. Funkcijska komponenta također vraća HTML i ponaša se na sličan način kao i klasična komponenta, ali funkcijske komponente mogu biti napisane s puno

manje koda i lakše se razumiju. (*React components*) Programski isječak 18 prikazuje *class* komponentu, dok 19 prikazuje njen ekvivalent, ali u obliku *function* komponente.

```
class Auto extends React.Component {
  render() {
    return <h2>Bok, ja sam auto!</h2>;
  }
}
```

Programski isječak 18. Klasna komponenta

```
function Auto() {
  return <h2>Bok, ja sam auto!</h2>;
}
```

Programski isječak 19. Funkcijska komponenta

Svaka upotreba ReactJS razvojne okoline obično uključuje sljedeće korake:

1. Korisnik razdvaja korisničko sučelje u hijerarhiju komponenata.
2. Statistička verzija se gradi u Reactu.
3. Minimalno zastupljenje stanja korisničkog sučelja (UI state) se identificira.
4. Identificira se gdje bi stanje trebalo biti smješteno (živjeti).
5. Obrnuti tok podataka se dodaje putem događaja i rukovatelja događajima (Minnick, 2023, p. 265).

Kada bi se neko web sučelje razdvojilo u hijerarhiju komponenata, to bi izgledalo ovako:

- Logo
- Polje za pretragu
- Gumb Prijava
- Traka za navigaciju
- Rezultati pretrage
- Pojedinačan rezultat pretrage
- Bočna traka (Minnick. 266)

Zatim treba stvoriti statičku verziju u Reactu. Nakon što su analizirane različite komponente koje čine korisničko sučelje, sljedeći korak u procesu je prevesti svaku od tih komponenata u statički kod. Ovaj korak možete zamisliti kao stvaranje HTML odlomka za svaku komponentu, uz dodatak Javascript funkcije koja generira taj HTML.(Minnick, 2023, p. 266)

Slijedi primjer statičke verzije navigacije, koji se može vidjeti u programskom isječku 20.

```
function TopNavBar() {
 return (
 <nav>
 <ul>
 <li>All</li>
 <li>News</li>
 <li>Videos</li>
 <li>Images</li>
 <li>Books</li>
 <li>More</li>
 </ul>
 </nav>
 );
}
export default TopNavBar;
```

Programski isječak 20. Statička navigacija

Sljedeći korak je definiranje stanja u Reactu, a stanje su zapravo podaci koji uzrokuju promjene u korisničkom sučelju. Primjer stanja može se dati kroz proučavanja primjera tražilice. U korisničkom sučelju najosnovniji dijelovi podataka su termin pretrage i rezultati pretrage. Od ova dva, onaj koji uzrokuje promjene u korisničkom sučelju je termin pretrage koji korisnik unese. Ostatak korisničkog sučelja jednostavno reagira na ovaj ključni podatak kako bi generirao rezultate pretrage, bočnu traku, reklame i druge elemente. Nakon što se odredi stanje, treba odlučiti u kojoj će komponenti to stanje biti smješteno. Razmatra se stanje kao privatno svojstvo objekta (jer to zaista jest). Samo komponenta koja posjeduje to svojstvo stanja može ga mijenjati, no vlasnik svojstva stanja može ga dijeliti s djecom kao vrijednost koja im se može čitati, ali nije dopušteno mijenjati je.(Minnick, 2023, p. 268)

Posljednji korak je slušanje događaja koji će uzrokovati promjenu stanja i proslijeđivanje povratnih funkcija (callback funkcija) djeci koja trebaju uzrokovati promjenu stanja. U primjeru pretraživača, slušač događaja može osluškivati događaj "submit" na elementu <input> u komponenti za pretragu. Kada se taj događaj detektira, rukovatelj događaja koristi povratnu

funkciju za ažuriranje podataka stanja u komponenti App s novom vrijednošću.(Minnick, 2023, p. 270) Što se tiče samog korištenja ReactJS razvojne okoline, potrebno je uzeti u obzir nekoliko faktora. Prvi je način rukovanja ReactJS komponentima:

- Zadatak React biblioteke je renderiranje stabla komponenata. Funkcionira slično tome kako renderni engine preglednika renderira DOM stablo iz HTML elemenata.
- Zadatak ReactDOM biblioteke je preuzeti stablo komponenata iz React biblioteke i koristiti ga za ažuriranje DOM-a u pregledniku.

Odgovornost ReactJS programera je izrada komponenata i njihovo renderiranje pomoću React biblioteke. Postupci promjene onoga što se prikazuje u prozoru preglednika automatski su upravljani od strane ReactDOM biblioteke. Drugi faktor povezan je s deklarativnom prirodom ReactJS razvojnog okruženja. Jedan od ključnih koncepta u Reactu jest njegova deklarativna priroda. Deklarativni sustav je onaj u kojem željeno stanje pruža programer kako bi sustav generirao, umjesto da se specificiraju konkretni koraci koje sustav mora poduzeti kako bi postigao te rezultate. Kako to funkcionira u Reactu jest da komponente koje se pišu opisuju kako će izgledati i ažurirati korisničko sučelje, bez potrebe za izravnom manipulacijom DOM-om kako bi se promijenilo ono što se prikazuje u pregledniku. Treći faktor odnosi se na Javascript temelje koje ReactJS razvojna okolina koristi. To je zato što je React sam po sebi prilično mala biblioteka za renderiranje stabla komponenata. Za razliku od nekih drugih Javascript biblioteka i okvira, React nema puno posebnih funkcija i alata koje morate naučiti kako biste je mogli koristiti. Reactovi programeri tvrde da je React "idiomatski Javascript", što znači da se pridržava načina na koji se piše Javascript, umjesto da stvara vlastiti način rada. (Minnick, 2023, p. 271)

Valja spomenuti i jednu od najnovijih značajka ReactJS razvojne okoline, a to je takozvano svojstvo istodobnosti. Ta istodobnost nije sama po sebi značajka. To je novi mehanizam koji omogućava Reactu da istovremeno pripremi više verzija korisničkog sučelja. Može se smatrati detaljem implementacije - njegova vrijednost leži u značajkama koje otključava. React može započeti proces ažuriranja, privremeno ga pauzirati te ga kasnije nastaviti. Može čak i napustiti ažuriranje koje je već u tijeku ako je potrebno. Ključna stvar je da React jamči da će korisničko sučelje ostati konzistentno, čak i ako se ažuriranje prekine. Kako bi to postigao, React odgađa izvođenje promjena na DOM-u dok ne procijeni cijelo stablo. Ta sposobnost omogućava Reactu da u pozadini pripremi nove zaslone bez ometanja glavne niti. Kao rezultat toga, korisničko sučelje može brzo reagirati na korisnički unos, čak i kada je u tijeku značajan zadatak renderiranja, što pruža besprijekorno korisničko iskustvo. Još jedan primjer odnosi se

na ponovnu upotrebu stanja. Moguće je privremeno ukloniti određene dijelove korisničkog sučelja sa zaslona i kasnije ih ponovno dodati, pri čemu se koristi prethodno stanje. Na primjer, kada korisnik prelazi između različitih zaslona i zatim se vraća na početni zaslon, React bi trebao imati mogućnost vraćanja prethodnog zaslona u istom stanju kao što je bio prije. (*REACT V18.0*)

3.2. Usporedba razvojnih okolina

ReactJS jedna je od najpopularnijih razvojnih okolina, no nije jedina. VueJS također je popularna razvojna okolina koja koristi Javascript. Iako je VueJS zasebna razvojna okolina, postoje neke zajedničke značajke koje dijeli sa ReactJS razvojnom okolinom:

- Vue.js ažurira preglednikov DOM na temelju promjena u posebnim vrijednostima u vašoj aplikaciji koje se nazivaju stanjem aplikacije.
- Vue stvara korisnička sučelja iz komponenata koje su napisane deklarativno.
- Vue koristi virtualni DOM (virtual DOM) kako bi izračunao razlike između onoga što vaša aplikacija renderira i što se nalazi u preglednikovom DOM-u.

Ove sličnosti pokazuju da i React i Vue dijele neke ključne koncepte i tehnike u izgradnji korisničkih sučelja.

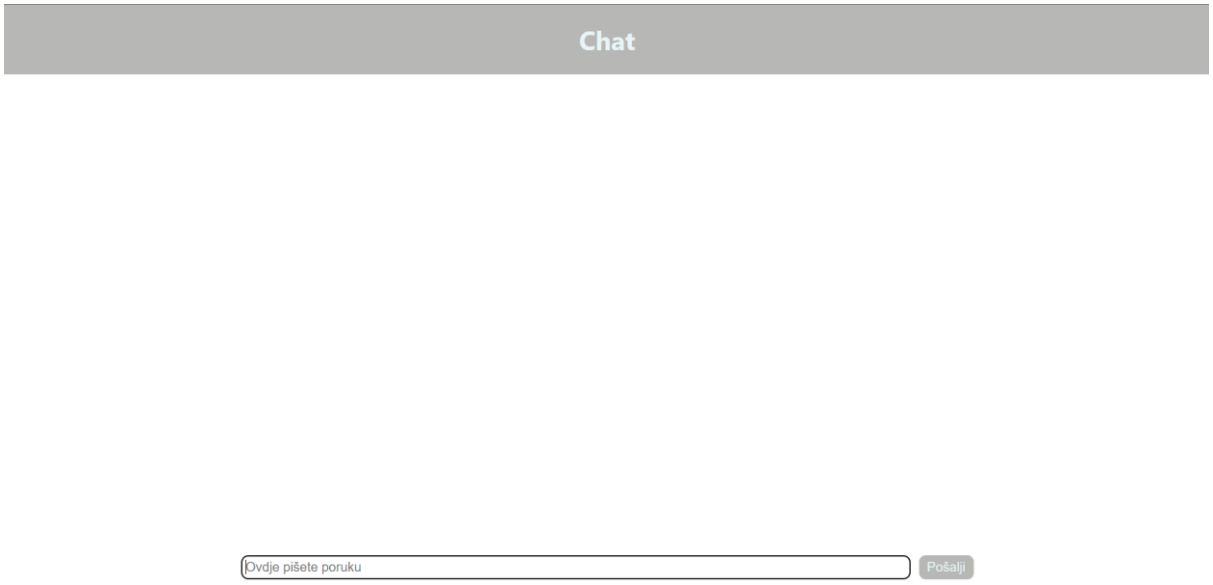
No, isto tako, postoje značajne razlike između navedenih razvojnih okolina:

- „Prije nego što se pretpostavi da je Vue.js isto što i ReactJS, važno je biti upućen u značajne razlike između ova dva, uključujući sljedeće:
- • Vue.js je postupno usvajajući (incrementally adoptable) - Vue.js komponente lako se mogu koristiti u drugim aplikacijama koje nisu bazirane na Vue.js-u. Iako je to moguće postići i s ReactJS-om, rijetko tko to radi.
- • Vue.js predlošci mogu se pisati koristeći običan HTML, za razliku od ReactJS-a koji zahtijeva upotrebu JSX-a.
- • Vue.js je okvir (framework), dok je ReactJS biblioteka (library). Kako će se kasnije vidjeti u ovom poglavlju, Vue.js pruža mnogo više strukture i ugrađenih funkcionalnosti u usporedbi s Reactom. U praksi, to znači da ReactJS pruža veću slobodu programerima, dok Vue.js nudi više jednostavnosti. (Minnick, 2023, p. 344)

4. Chat Aplikacija

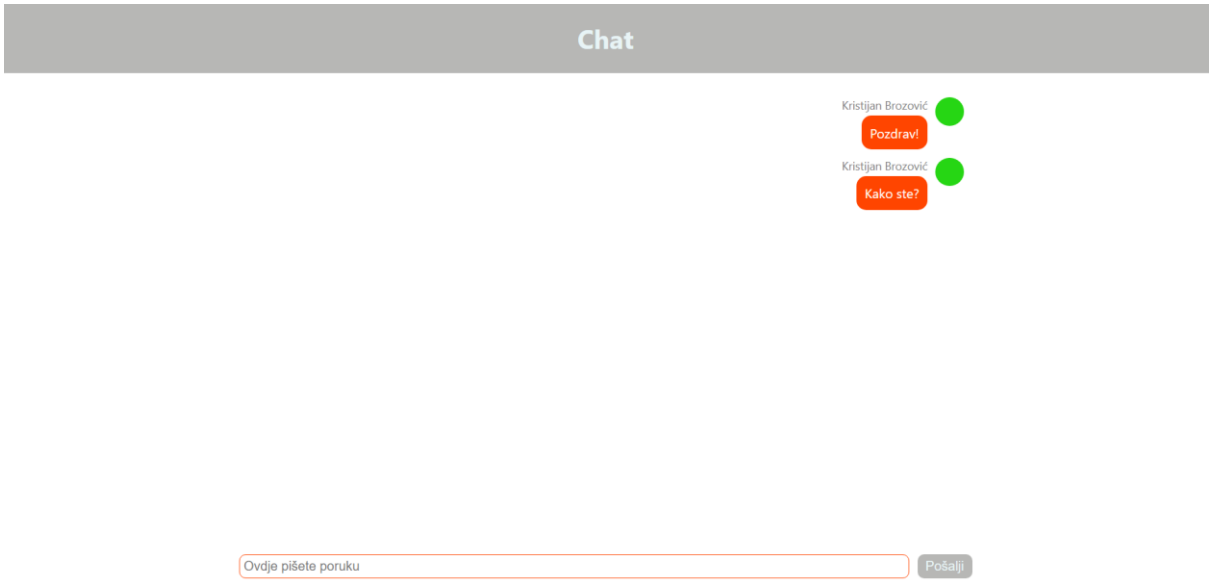
U svrhu dobivanja uvida u praktični dio korištenja HTML-a, CSS-a, Javascript-a i React-a, napravljena je chat aplikacija. Aplikacija služi za komunikaciju više korisnika. Svakom se korisniku automatski dodijeli nasumično odabrano ime koje se zatim kombinira sa nasumično odabranim prezimenom. Također, svakom se korisniku dodijeli nasumično odabrana boja koja služi za raspoznavanje različitih korisnika.

Na slici 1 vidljivo je početno stanje chat aplikacije. Aplikacija nije pretjerano dotjerana, što znači da je razina CSS oblikovanja dosta niska. Početni zaslon aplikacije sastoji se od sivog zaglavlja, koji u sebi sadrži naslov '*Chat*', a u podnožju aplikacije nalazi se traka u kojoj se piše poruka zajedno sa gumbom za slanje te poruke.



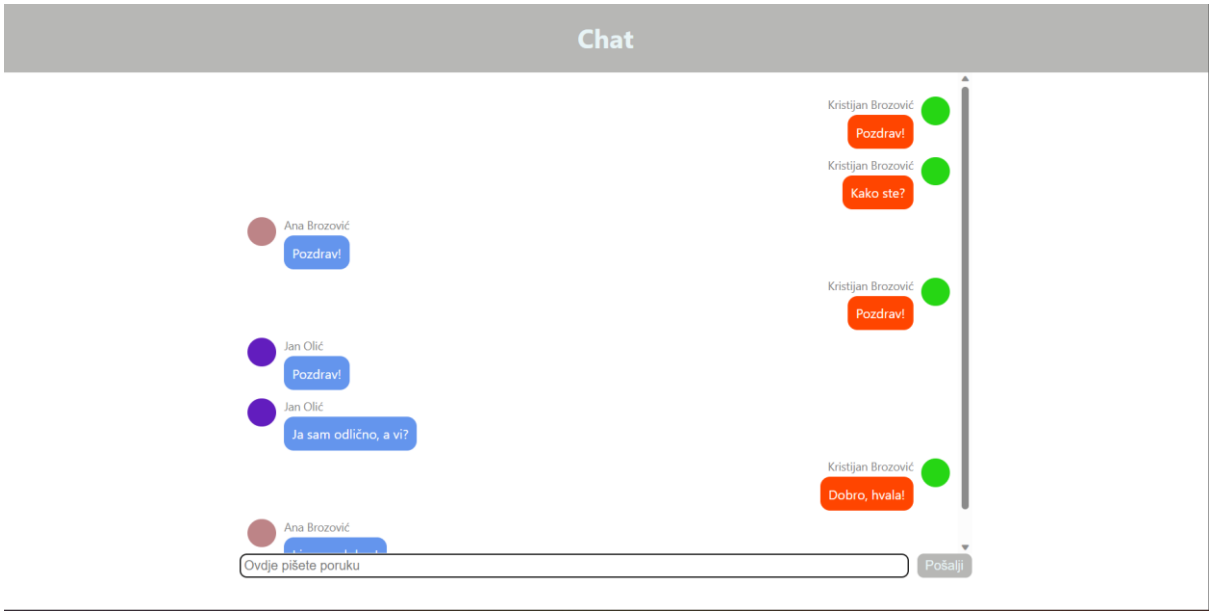
Slika 1. Početno stanje chat aplikacije

Nadalje, u slici 2 vidi se stanje chat aplikacije kada samo jedan korisnik šalje poruke. Korisnik u traku za pisanje poruka napiše sadržaj svoje poruke, a nakon što pritisne gumb '*Pošalji*', poruka nestaje iz trake i pojavljuje se u bijelom prostoru ispod sivog zaglavlja aplikacije. Kao što se vidi na slici, korisniku je nasumično dodijeljena boja, zelena, ali i ime i prezime, odnosno '*Kristijan Brozović*'.



Slika 2. Aplikacija nakon slanja poruke trenutnog korisnika

Zatim, može se simulirati razgovor više korisnika koristeći ovu chat aplikaciju. Simulira se na način da se otvori više kartica u web pregledniku sa adresom aplikacije. Kada se to učini, svaka otvorena kartica je zapravo jedan jedinstveni korisnik. Na slici 3 prikazan je razgovor više korisnika koristeći više otvorenih kartica web preglednika. Svaki korisnik dobio je drugačiju, nasumično odabranu boju, ali i ime i prezime. Naprimjer, korisnik '*Jan Olić*' dobio je plavu boju za razliku od prvog korisnika kojem je dodijeljena zelena boja. Trenutni korisnik svoje poruke vidi smještene u narančaste oblike, dok su poruke ostalih korisnika smještene u oblike plave boje.



Slika 3. Razgovor više korisnika koristeći chat aplikaciju

Aplikacija se sastoji od sljedećih datoteka: index.html, index.js, index.css, App.js, App.css, Input.js i Messages.js. Svaka od datoteka pridonosi nekom od aspekata chat aplikacije. HTML datoteke, odnosno index.html, predstavlja sučelje, odnosno sadržaj - u toj se datoteci prikazuju sve ostale datoteke. App.css i index.css sadrže upute za oblikovanje stila chat aplikacije – boje, položaj, izgled i dimenzije komponenata koji se prikazuju na sučelju. U ostalim datotekama, odnosno Javascript datotekama, razvija se sva logika i funkcionalnost aplikacije. Unutar ovih datoteka nalaze se komponente koje se povezuju i zajedno čine potpunu aplikaciju. Index.js ima posebnu ulogu – to je datoteka u kojoj se spajaju sve ostale datoteke, a zatim 'šalju' u index.html gdje se prikazuju.

Index.html

Index.html sastoji se od dva dijela: zaglavlje, koje je označeno početnim i završnim tagovima <head> i </head> te tijelom datoteke, koje se nalazi između tagova <body> i </body>. Unutar zaglavlja nalaze se metapodaci kao što su poveznica sa API uslugom, koja u slučaju ove aplikacije ima ulogu servera, veličinu sadržaja ovisno dimenzijama ekrana korisnika te naslov. Poveznica sa API uslugom nalazi se unutar taga <script>. Unutar tagova <body> i </body> nalazi se sadržaj sučelja. Sadržaj sučelja, barem u ovoj datoteci, je prazan element: “Oznaka definira podjelu ili sekciju u HTML dokumentu. Oznaka se koristi kao kontejner za HTML elemente - koji se potom stiliziraju pomoću CSS-a ili manipuliraju pomoću Javascripta. Oznaka lako se stilizira korištenjem atributa class ili id. Unutar oznake može se staviti bilo kakav sadržaj! (*HTML div tag*) Unutar početnog 'taga', nalazi se id sadržaja 'root'. Atribut 'id' služi kako bi se index.js povezao sa upravom ovim elementom: „Atribut 'id' određuje jedinstveni identifikator za HTML element. Vrijednost atributa 'id' mora biti jedinstvena unutar HTML dokumenta.(HTML-the id attribute) Odnosno, to znači da se sadržaj iz datoteke index.js prikazuje u ovom elementu.

Cijeli sadržaj index.html datoteke izgleda ovako:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <script src="https://cdn.scaledrone.com/scaledrone.min.js"></script>
```

```
    <meta charset="utf-8" />
    <meta
      name="viewport"
      content="width=device-width, initial-scale=1, shrink-to-fit=no"
    />
    <title>Chat App</title>
  </head>
  <body>
    <div id="root"></div>
  </body>
</html>
```

Messages.js

U datoteci Messages.js nalazi se komponenta Messages koja sadrži funkciju render.:

```
class Messages extends Component {
  render() {
    const { messages } = this.props;
    return (
      <ul className="Mess-list">
        {messages.map((m) => this.renderMessage(m))}
      </ul>
    );
  }
```

Unutar funkcije render, komponenta prima poruku koju treba prikazati. Zatim se stvara lista koja sadrži podatke kao što su ime i avatar pošiljatelja, ali i sadržaj poruke. U ovoj komponenti također se nalazi i funkcija renderMessaage:

```
renderMessage(message) {
    const { member, text } = message;
    const { currentMember } = this.props;
    const messageFromMe = member.id === currentMember.id;
```

```
  const className = messageFromMe ? "Mess-mess currentMember" : "Mess-mess";
  return (
    <li className={className} key={this.getKey()}>
      <span
        className="avatar"
        style={{ backgroundColor: member.clientData.color }}
      />
      <div className="Mess-content">
        <div className="username">{member.clientData.username}</div>
        <div className="text">{text}</div>
      </div>
    </li>
  );
}
```

Ova funkcija služi za prikazivanje pojedine poruke.

Input.js

U Input.js datoteci nalazi se komponenta 'Input' koja sadrži tekstualno polje i gumb koji šalje taj tekst. Trenutni uneseni tekst prati se kroz 'state':

```
class Input extends Component {
  state = {
    text: "",
  };
```

Komponenta također sadrži funkciju render:

```
render() {
    return (
      <div className="Input">
        <form onSubmit={(e) => this.onSubmit(e)}>
          <input
            onChange={(e) => this.onChange(e)}
```

```
          value={this.state.text}
          type="text"
          placeholder="Ovdje pišete poruku"
          autoFocus={true}
        />
        <button>Pošalji</button>
      </form>
    </div>
  );
}
```

Kada se vrijednost promijeni, 'state' reflektira tu promjenu.

```
onChange(e) {
  this.setState({ text: e.target.value });
}

onSubmit(e) {
  e.preventDefault();
  this.setState({ text: "" });
  this.props.onSendMessage(this.state.text);
}
```

App.js

App.js datoteka služi za slanje i primanje poruka, ali i igra ulogu kod prikaza komponenata na sučelju. Sadržaj App.js datoteke započinje s četiri 'import' akcije:

```
import React, { Component } from "react";
import "./App.css";
import Messages from "./Messages";
import Input from "./Input";
```

Moguće je uvoziti/izvoziti React komponente i to je jedna od osnovnih operacija koje se obavljaju. U Reactu se koristi ključna riječ 'import' i 'from' za uvoz određenog modula ili imenovanog parametra.(*ReactJS Importing and exporting*)

Dakle, 'import' se koristi kako bi se sadržaj jedne datoteke, ili ReactJS knjižnice, 'uveo' u željenu datoteku gdje se zatim može i koristiti. Zatim, slijedi funkcija pod nazivom 'randomName':

```
function randomName() {
  const imena = [
    "Karlo",
    "Valentina",
    "Ivona",
    "Dejan",
    "Danica",
    "Stjepan",
    "Gordan",
    "Jan",
    "Kristijan",
    "Lucija",
    "Mihael",
    "Maja",
    "Ana",
    "Sandra",
    "Mateja",
    "Nikola",
    "Matija",
    "Tomislav",
    "Ivan",
    "Branko",
    "Zlatan",
    "Duje",
  ];
  const prezimena = [
    "Huljak",
    "Modrić",
    "Kovačić",
    "Brozović",
```

```
    "Mandžukić",
    "Olić",
    "Lovren",
    "Vlašić",
    "Pašalić",
    "Kramarić",
    "Prosinečki",
    "Šuker",
    "Oršić",
    "Vida",
    "Perišić",
  ];
  const ime = imena[Math.floor(Math.random() * imena.length)];
  const prezime = prezimena[Math.floor(Math.random() * prezimena.length)];
  return ime + " " + prezime;
}
```

Unutar funkcije 'randomName', nalaze se dvije liste. Prva lista sadrži imena, a druga lista sadrži prezimena. Zatim, u varijabli 'ime', pomoću metoda 'Math.floor' i 'Math.random' nasumično se odabere jedno ime iz liste 'imena'. Isti postupak se ponovi i za prezimena, te se vrijednost spremi u varijablu 'prezime'. Dakle, rezultat ove funkcije je jedno nasumično odabrano ime zajedno sa nasumično odabranim prezimenom, npr. Ivan Šuker. Sličnim postupkom nasumično se odabire i boja:

```
function randomColor() {
  return "#" + Math.floor(Math.random() * 0xffffff).toString(16);
}
```

Te dvije funkcije, randomName i randomColor, koriste se u klasi App:

```
class App extends Component {
  state = {
    messages: [],
    member: {
```

```
    username: randomName(),
    color: randomColor(),
  },
};
```

Dakle, unutar klase App se korisniku dodjeljuje nasumično odabrano ime i nasumično odabrana boja. Zasad, 'messages', odnosno poruka, ostaje prazna lista, no za popunjavanje te liste će se koristiti posebna komponenta pod nazivom Messages.js. Osim dodjeljivanja imena i boje korisniku, App klasa definira sučelje i funkcionalnost chat aplikacije:

```
render() {
  return (
    <div className="App-app">
      <div className="header">
        <h1>Chat</h1>
      </div>
      <Messages
        messages={this.state.messages}
        currentMember={this.state.member}
      />
      <Input onSendMessage={this.onSendMessage} />
    </div>
  );
}
```

Dakle, ovaj dio App klase sastoji se od dva elementa, a svaki ima atribut – klasu koja se koristi u App.css datoteci za stilsko oblikovanje sučelja chat aplikacije. Također, tu se nalazi <h1> element, odnosno naslov koji će se prikazivati na sučelju, u ovom slučaju 'Chat'. Nadalje, tu se nalaze i dvije komponente: Messages i Input koje su na početku uvedene u datoteku pomoću 'Import' akcije. Nadalje, u App klasi se aplikacija povezuje sa uslugom Scaledrone, koja omogućuje komunikaciju između korisnika u stvarnom vremenu:

```
constructor() {
  super();
  this.drone = new window.Scaledrone("wIlgk610Cm5O0NUe", {
    data: this.state.member,
  });
  this.drone.on("open", (error) => {
    if (error) {
      return console.error(error);
    }
    const member = { ...this.state.member };
    member.id = this.drone.clientId;
    this.setState({ member });
  });
  const room = this.drone.subscribe("observable-room");
  room.on("data", (data, member) => {
    const messages = this.state.messages;
    messages.push({ member, text: data });
    this.setState({ messages });
  });
 }
```

Ovaj dio sintakse stvara novu instancu Saldrone-a i prenosi podatke o korisniku koji trenutno koristi uslugu. Zatim se aplikacija spaja sa sobom, odnosno grupom korisnika koji međusobno mogu slati poruke: const room = this.drone.subscribe("observable-room");. Zatim se sadržaj poruke i podaci klijenta dodaju u 'state'. Na kraju, funkcija onSendMessage šalje poruku svim korisnicima u sobi:

```
onSendMessage = (message) => {
  this.drone.publish({
    room: "observable-room",
    message,
  });
 };
```

Index.js

Kao što je već i spomenuto, index.js datoteka služi kao poveznica između App.js, odnosno same aplikacije, i index.html datoteke, koja služi kao izvor korisničkog sučelja. Dakle, prvo se u datoteku uvode sve relevantne komponente:

```
import React from "react";
import ReactDOM from "react-dom";
import "./index.css";
import App from "./App";
```

S obzirom da App.js komponenta sadrži i Input.js i Messages.js komponente, index.js indirektno sadrži sve komponente u ovoj chat aplikaciji, a zatim ih prikazuje preko index.html datoteke:

```
ReactDOM.render(<App />, document.getElementById("root"));
```

5. Zaključak

Razvijanje web sučelja kompleksan je proces koji zahtijeva poznavanje i savladavanje različitih tehnologija. HTML koristi se za prikaz sadržaja i pomoću 'tagova' dodjeljuje različite razine 'važnosti' dijelovima tog sadržaja. CSS se koristi za stilsko obilježavanje tog sadržaja te određuje koju će poziciju određeni dio sadržaja zauzimati na korisničkom sučelju. Javascript pak daje cijelom web sučelju funkcionalnost. Javascript je možda i najteži za savladati jer je to programski jezik u punom smislu – potrebno je savladavanje koncepata kao što su nizovi, objekti, funkcije i metode, a zatim i interakciju između tih koncepata. ReactJS razvojna je okolina, što znači da spaja sve nabrojene tehnologije u jedno. ReactJS vidi korisničko sučelje kao skup komponenata, a svaki od komponenata sastoji se od HTML, CSS, ili Javascript sintakse. No, ReactJS, ima posebnu sintaksu koja je bazirana na Javascriptu, tako da je potrebno naučiti osnovne tehnologije kako bi se mogao koristiti. Iako je ReactJS jedna od najpopularnijih razvojnih okolina, postoje i druge, kao što je VueJS. Ove dvije razvojne okoline imaju neke zajedničke značajke, no postoje i velike razlike. ReactJS koristio se za izradu chat aplikacije opisane u ovom radu. Aplikacija funkcionira zahvaljujući usluzi 'Scaldrone', koja omogućuje slanje i primanje poruka. Aplikacija svakom novom korisniku dodjeljuje nasumično odabrano ime i prezime te nasumično odabranu boju. Korisnici mogu izmjenjivati poruke, a svaka poruka ostaje zapisana u korisničkom sučelju.

6. Literatura

1. GeeksforGeeks (2021) “ReactJS Importing and exporting” GeeksforGeeks. Dostupno na: https://www.geeksforgeeks.org/reactjs-importing-exporting/.
2. CSS Syntax. Dostupno na: https://www.w3schools.com/Css/css_syntax.asp.
3. Getting started with React - Learn web development | MDN (2023). Dostupno na: https://developer.mozilla.org/en-US/docs/Learn/Toolsandtesting/Client-sideJavascriptframeworks/Reactgettingstarted
4. HTML div tag. Dostupno na: https://www.w3schools.com/Tags/tag_div.asp.
5. HTML- The id attribute. Dostupno na: https://www.w3schools.com/htmL/html_id.asp.
6. JavaScript methods. Available at : https://www.w3schools.com/js/js_object_methods.asp.
7. Kantor, I. Arrays. Dostupno na: https://Javascript.info/array.
8. Minnick, C. (2023) *JavaScript All-in-One For Dummies*
9. React components. Dostupno na: https://www.w3schools.com/react/react_components.asp.
10. REACT V18.0 – REAct (2022). Dostupno na: https://react.dev/blog/2022/03/29/react-v18.
11. SPA (Single-page application) - MDN Web Docs Glossary: Definitions of Web-related terms | MDN (2023). Dostupno na: https://developer.mozilla.org/en-US/docs/Glossary/SPA.
12. Tittel, E. and Burmeister, M. (2008) HTML 4 for Dummies. John Wiley & Sons.

7. Popis programskih isječaka

8. Popis slika

Osnove razvoja web sučelja i izrada chat aplikacije

Sažetak

U ovom radu, opisat će se proces stvaranja jednostavnije chat aplikacije pomoću ReactJS razvojne okoline. Kako je ReactJS razvojna okolina koja se najčešće koristi u svrhu stvaranja SPA(Single-page application) projekata, dobar je odabir za ovakav projekt. Za rad u ReactJS-u potrebno je poznavanje HTML-a, CSS-a i Javascript-a, pa će se prije opisivanja rada u React-u, opisati ti programski jezici, odnosno njihovi koncepti koji se koriste u izradi same aplikacije. Tags u HTML-u, class i id u CSS-u te funkcije i liste u Javascript-u su neki od pojmova koji će se opisati i objasniti u ovom radu. Zatim će se predstaviti način rada i koncepti u širem smislu vezano uz ReactJS – kako se HTML, CSS i Javascript koriste u izradi komponenata koji zajedno čine funkcionalnu aplikaciju. Potom će se opisati postupak izrade same aplikacije. Aplikacija će se moći testirati pomoću više otvorenih kartica u web pregledniku gdje će se u svakoj novoj kartici generirati novi korisnik sa nasumično odabranim imenom i prezimenom, kao i jedinstvenom, nasumično odabranom bojom koja će poslužiti umjesto profilne slike korisnika. Kako bi aplikacija bila u potpunosti funkcionalna, moralo bi se izaći iz sfere front-enda, odnosno stvaranja korisničkog web sučelja. Iz tog razloga, kao back-end koristit će se usluga ‘Scaledrone’ koja će omogućiti rad ove aplikacije.

Ključne riječi: HTML, CSS, Javasript, ReactJS, chat aplikacija

Basics of front-end development and creating a chat application

Summary

This work will describe the process of creating a simpler chat application using the ReactJS development environment. Since ReactJS is a development environment most commonly used for creating Single-Page Application (SPA) projects, it is a good choice for such a project. Working with ReactJS requires knowledge of HTML, CSS, and JavaScript, so before describing the work in React, we will explain these programming languages and their concepts used in building the application. Terms like HTML tags, class and id in CSS, as well as functions and lists in JavaScript, are some of the concepts that will be described and explained in this work. We will then introduce the working principles and concepts related to ReactJS in a broader sense - how HTML, CSS, and JavaScript are used to create components that together form a functional application. Next, the process of building the application itself will be described. The application will be testable by opening multiple tabs in a web browser, where each new tab will generate a new user with randomly selected first and last names, as well as a unique, randomly chosen color to be used instead of a user profile picture. To make the application fully functional, it will be necessary to go beyond the front-end realm, i.e., creating the user interface. For this reason, the "Scaledrone" service will be used as the back end, enabling the operation of this application.

Key words: HTML, CSS, Javasript, ReactJS, chat application