Dinamička vizualizacija podataka na webu

Vinko, Patrik

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Patrik Vinko

DINAMIČKA VIZUALIZACIJA PODATAKA NA WEBU

ZAVRŠNI RAD

Varaždin, 2022.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Patrik Vinko

JMBAG: 0016141446

Studij: Informacijski sustavi

DINAMIČKA VIZUALIZACIJA PODATAKA NA WEBU

ZAVRŠNI RAD

Mentor/Mentorica:

Izv. prof. dr. sc. Dijana Plantak Vukovac

Varaždin, rujan 2022.

Patrik Vinko

Izjava o izvornosti

Izjavljujem da je moj završni rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

_____

Sažetak

Ovaj rad razrađuje i objašnjava važnost vizualizacije podataka, sve njene aspekte te pravila koja se s tim pojmom asociraju. Glavna tehnologija koja je korištena u ovom radu jest JavaScript-ova biblioteka D3. D3 biblioteka je namijenjena kako bi se olakšao prikaz podataka te se ujedno omogućila interaktivnost korisnika s web-stranicom i podacima na njoj. Također je objašnjena manipulacija DOM-om (engl. *Document Object Model*) koja je usko vezana sa prikazivanjem podataka. Prikazane su osnove crtanja te prikaza podataka. D3 ima veoma zanimljivu i intuitivnu sintaksu pa se uz nju može lakše doći do već prije zacrtanog rješenja.

U praktičnom dijelu rada prikazana je karta Republike Hrvatske sa svim njenim županijama te je za svaku od županija prikazana vremenska prognoza. Vremensku prognozu se može vidjeti po satima u danu. Klikom miša na određenu županiju, prikazuje se skočni prozor sa više informacija o vremenskim uvjetima kao što su temperatura, brzina vjetra, vlažnost,…

Konačni rezultat predstavlja funkcionalnu i *easy-to-use* aplikaciju za pregled vremena sa nekoliko dodatnih funkcionalnosti koje će privući korisnikovu pozornost. Teorija, zajedno s praktičnim radom, u potpunosti objašnjava svaki bitni aspekt vizualizacije podataka na webu i promatra ju kao prošlost, sadašnjost i budućnost kod razvoja web stranica.

Ključne riječi: vizualizacija, podaci, DOM, D3, karta, dizajn, prognoza, API

Sadržaj

1. Uvod 1
2. Vizualizacija podataka 2
   2.1. Tablice 2
   2.2. Složeni stupčasti grafikon 3
   2.3. Kružni grafikon 3
   2.4. Linijski dijagram 4
   2.5. Histogram 5
   2.6. Toplinska karta 5
   2.7. Mapiranje slika (tree maps) 6
3. Pravila prikaza podataka s aspekta dizajna 7
   3.1. Balansiranje dizajna 7
   3.2. Naglašavanje bitnih aspekata 8
   3.3. Jednostavnost 9
   3.4. Pametno korištenje uzoraka 9
   3.5. Proporcije 10
   3.6. Točan ritam 10
   3.7. Raznolikost 11
   3.8. Tema 11
4. Vizualizacija podataka na webu 12
   4.1. Statička i dinamička vizualizacija 13
   4.2. Alati za dinamičku vizualizaciju 13
5. Manipulacija DOM-om 15
   5.1. DOM 15
   5.2. Manipulacija 16
   5.3. Document (Dokument) 17
   5.4. Node (Čvor) 17
   5.5. Element 18
   5.6. NodeList 18
   5.7. Attr 19
   5.8. NamedNodeMap 19
6. D3.js biblioteka 20
   6.1. Selekcije 20
   6.2. Dinamička svojstva 21
   6.3. Enter i Exit selekcije 21
   6.4. Tranzicije 21
   6.5. D3 Geo 22
   6.6. Osnove crtanja 22
7. Praktičan dio rada 23
   7.1. Opis postupka 23
   7.2. Konačna dinamička vizualizacija 32
8. Zaključak 33
9. Literatura 34
10. Popis slika 37

1. UVOD

Statistika je grana matematike koja se bavi sakupljanjem, analizom, interpretacijom te pregledom podataka. U čovječjoj je naravi da zapisuje i pregledava ono što je zapisao. Nakon nekog vremena, podaci koje je sakupio, postanu preglomazni pa samim time i teže čitljivi. Kako bi se riješio taj problem, ljudi su počeli vizualizirati podatke pomoću raznih dijagrama, grafikona, karta ili tablica. Vizualizacija podataka bi se mogla svrstati kao moderniji razvoj statistike, zajedno sa statističkom grafikom, ali kroz svoju povijest je dobivala razna tehnološka poboljšanja, zbog kojih se danas toliko puno upotrebljava. Neka od tih tehnoloških poboljšanja su tehnologije za crtanje i reprodukciju slika, česti napredci u matematici, ili točnije, u statistici te razna poboljšanja u prikupljanju podataka. Podaci ne daju puno informacija, ne prikazuju cijelu sliku oko toga što žele prikazati. Njihov potencijal se upotpunjuje korištenjem raznih alata za vizualizaciju. Kako bi se lakše prikazala važnost vizualizacije, zamislimo ovo, naša baza, web-stranica ili obična tablica sadrži pregršt podataka. Naš izvor s podacima se sastoji od vremenskih uvjeta za svaku državu. Sve što mi želimo dobiti je prosječna temperatura za svaku državu. Umjesto da čitamo podatke tako dugo dok ne dođemo do određene države, te podatke možemo vizualizirati npr. pomoću toplinske karte, ili obične karte na kojoj su napisane temperature. Također, u podacima se mogu nalaziti informacije koje su teške za razumjeti. Neke od tih informacija su vrijednosti koeficijenata (npr. trenja, korelacije, smjera pravca,…). Nama te vrijednosti ne znače ništa tako dugo dok ih ne stavimo u omjer sa drugim vrijednostima. To se najbolje može postići uz pomoć vizualizacije.

Rad se sastoji od 9 poglavlja. Drugo poglavlje će govoriti o onom što je zapravo u centru ovog rada, vizualizacija podataka. Treće poglavlje govori o pravilima prikaza podataka s aspekta dizajna. Četvrto poglavlje povezuje vizualizaciju podataka i web, odnosno, objašnjava njihovu interakciju. U petom poglavlju će biti prikazana manipulacija DOM-om, odnosno objektnim modelom dokumenta koji označava sučelje s HTML dokumentom u strukturi sličnoj stablu. Šesto poglavlje se bavi alatom bez kojeg bi praktični dio bio nekoliko puta teži, D3.js biblioteka. Sedmo poglavlje je namijenjeno za objašnjavanje praktičnog dijela, prikaz problematike te konačnog rješenja. U osmom poglavlju se donosi zaključak ovog rada dok će deveto i deseto prikazivati literaturu odnosno popis slika i tablica.

2. VIZUALIZACIJA PODATAKA

Vizualizacija podataka je grafička reprezentacija podataka koji su nam dani. Vizualizacijom podataka se u čitljivom obliku prikazuju podaci, trendovi te odstupanja kod nekog seta podataka. Za takav prikaz, najviše se upotrebljavaju dijagrami, grafikoni te karte. Vizualizacija podataka jedan je od koraka kod procesa u znanosti o podacima (engl. *data science*), koji navodi da nakon što su podaci prikupljeni, obrađeni i modelirani, moraju se vizualizirati kako bi se mogle donijeti bitne odluke ili zaključci. [1]

Tipovi vizualizacije:

- Tablice
- Kružni grafikon i složeni stupčasti grafikon
- Linijski dijagram
- Histogram
- Toplinska karta
- Mapiranje slika

2.1. Tablice

Tablice se sastoje od redaka i stupaca pomoću kojih uspoređujemo podatke u svakoj pojedinoj ćeliji. Prikazuju podatke veoma strukturirano. Povećanjem podataka, smanjuje se čitljivost. [2]

Tablice koristimo kada podatke koje želimo prezentirati, ne možemo lako prikazati vizualno. Pomoću njih, možemo vidjeti preciznije vrijednosti, u kontrastu s grafikonima koji služe za približnu reprezentaciju podataka. Osim toga, tablice su lakše za razumjeti ukoliko skupovi podataka imaju više od dvije dimenzije. Vizualizacija grafikonom u tom slučaju postaje puno složenija. [30]

Znajući sve ovo, možemo s lakoćom razumjeti zašto se u znanstvenim radovima ili medicinskim izvještajima koriste tablice. Medicinski izvještaji su osobito osjetljiviji zbog toga što svi podaci (npr. o pacijentima, o lijekovima,…) moraju biti točni zbog toga što ljudski životi ovise o njima. [31]

	Bank loan monthly payments	Monthly lease payment	Minimum downpayment for lease	Total interest paid over 48 months	Monthly insurance payment
Ford Fusion	552	395	0	2,529	180
Honda Civic	538	424	0	2,466	236
Mazda 3	506	478	1,000	2,318	251
Toyota Yaris	435	490	1,000	1,992	198
VW Golf	596	550	2,500	2,730	244

Slika 1. Primjer tablice za vizualizaciju podataka[14]

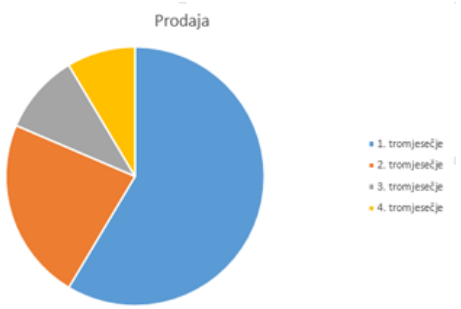
2.2. Složeni stupčasti grafikon

Složeni stupčasti grafikon u jednom stupcu reprezentira sve kategorije te se lako može vidjeti omjer između njih. Vrijednost na horizontalnoj osi je najčešće vrijeme. Ovakav grafikon je podijeljen u sekcije koje prikazuju dijelove cjeline. Pruža jednostavan način za prikazivanje i organizaciju podataka. Kod takvih grafikona se uspoređuju veličine određene sekcije te se tako iščitavaju podaci. Ovakav grafikon je najbolje koristiti ukoliko želimo prikazati dva različita seta podataka (npr. udio kupnje novijeg i skupljeg telefona u odnosu na stariji i jeftiniji kroz 12 mjeseca u intervalu od jedne godine). [3]

Slika 2. Primjer složenog stupčastog grafikona [15]

2.3. Kružni grafikon

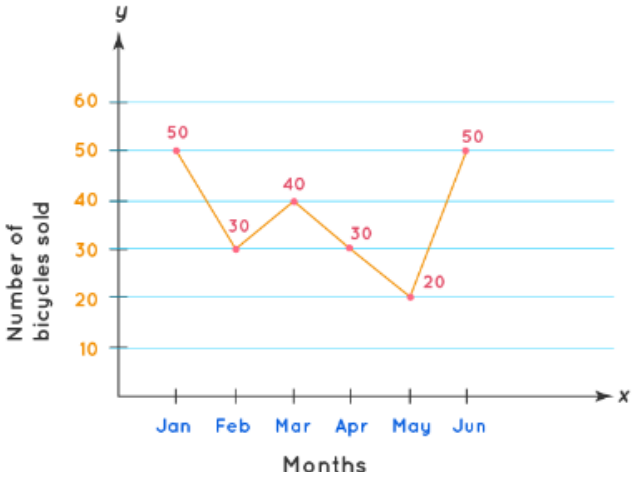
Za razliku od složenog trakastog grafikona, kružni grafikon je podijeljen na kružne isječke. Još jedna razlika između ova dva grafikona je ta što se u kružnom prikazuje omjer sekcija na kraju jednog određenog vremenskog intervala, dok stupčasti grafikon može prikazivati omjere u jednom vremenskom intervalu kroz više perioda (npr. kvartali). Kružni grafikon može postati koristan kada želimo naglasiti neki dio cjeline. Jedna od prednosti kružnog grafikona su te što mogu prikazivati puno više kategorija od složenog trakastog grafikona. Naime, u složenom trakastom grafikonu je mnogo složenije napraviti takvu reprezentaciju podataka te se samim time smanjuje čitljivost i razumljivost. [3]



Slika 3. Primjer tortnog grafikona [16]

2.4. Linijski dijagram

Ovaj dijagram prikazuje promjenu brojnosti jednog ili više entiteta kroz vrijeme. Koristi linije kako bi bolje prikazao te promjene. Te linije samo povezuju točke koje pak prikazuju prethodno definirane podatke (npr. podatke dobivene ispunjavanjem ankete). Najčešće, horizontalna os prikazuje vrijeme, dok vertikalna os prikazuje brojnost (kvantitativni podatak). Linijski graf najbolje prikazuje povećanje ili smanjenje nekog trenda. Ovakvi dijagrami se najčešće prikazuju na kraju godine kako bi lakše vizualizirali numeričke podatke. Kod njih, direktor neke tvrtke može lakše vidjeti kako mu se kroz mjesece prodavao glavni proizvod ili kako uopće njegova tvrtka posluje, tj. da li se vidi rast ili pad. Osim toga, u ovakvom se dijagramu lakše vide anomalije i odstupanja od nekog predodređenog standarda. Također, više linija može biti povućeno kako bi se vidjele promjene, odnosno usporedbe kod prodaje više različitih proizvoda što će nam koristiti u daljnjoj strategiji poslovanja. [4]

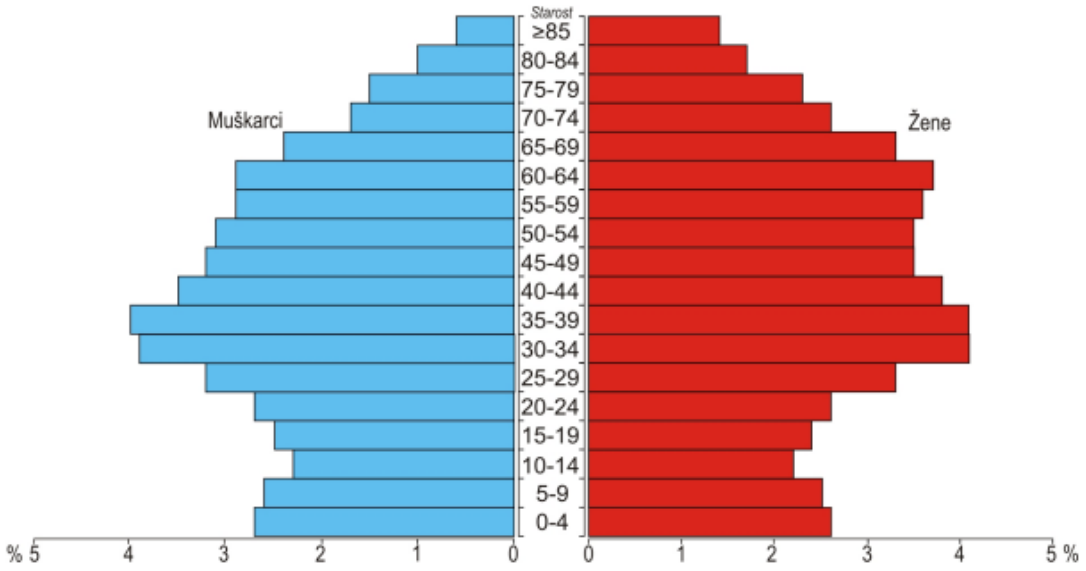


Slika 4. Primjer linijskog dijagrama [17]

2.5. Histogram

Ovaj grafikon prikazuje distribuciju nekih kvantitativnih podataka pomoću trakastog dijagrama te kod njega nema razmaka između stupaca. Predstavlja količinu podataka koji spadaju pod neki raspon (npr. broj godina kod korisnika). Kod takvog grafikona se lako mogu prepoznati određena odstupanja od većine.

Histogram se koristi kod numeričkih podataka, najčešće kod prikazivanja podataka o populaciji neke države. To može uključivati broj godina, visinu, težinu,… Najpoznatiji takav histogram je dobno-spolna piramida koja prikazuje 3 bitna podatka o populaciji neke države. Na glavnoj osi (u fokusu piramide) se nalaze rasponi godina, dok su ostala dva podatka spol te koliki postotak određena populacija zauzima prema ukupnoj populaciji države.

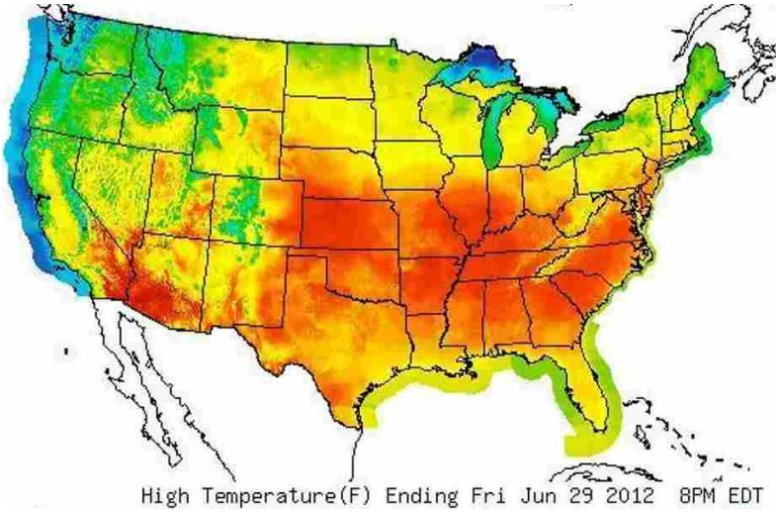


Slika 5. Primjer dobno-spolne piramide [18]

2.6. Toplinska karta

Ovaj grafički prikaz je veoma koristan kod vizualiziranja podataka po lokaciji. Lokacija može biti na karti, web stranici, nogometnom terenu,… Prikazuju koje lokacije su najkorištenije (engl. *hotspots*), te su takve prebojane toplim bojama (crvena, žuta,…). Dok one manje korištene su obojane hladnijim bojama (npr. plavom, zelenom). Toplinske karte su sve više zastupljenije na web stranicama. Daju uvid u količinu aktivnosti koje se mogu manifestirati na razne načine (npr. klikom na miš). Prema tome se može odrediti najkorištenija funkcionalnost, najprodavaniji proizvod, najiskorištenije područje za listanje po dokumentu itd. Toplinske karte

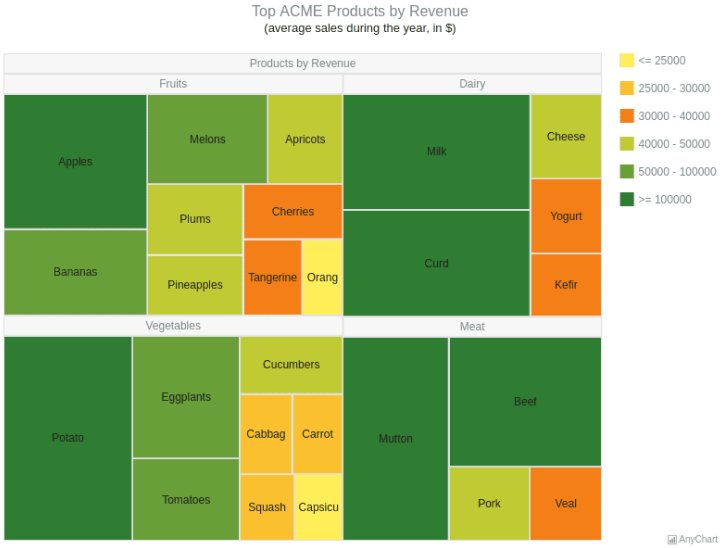
stoga mogu jako pomoći kod redizajniranje nekih elemenata u dokumentu. One ujedno reprezentiraju ponašanje jednog ili više korisnika kod korištenja stranice i šalju povratne informacije bez da korisnik uopće zna o tome. Jedan od boljih alata koji stvara takve toplinske karte je Hotjar. [32]



Slika 6. Primjer toplinske karte [19]

2.7. Mapiranje slika (engl. *tree maps*)

Prikazuje hijerarhijske podatke kao setove ugniježđenih oblika (najčešće pravokutnika). Mapiranje slika korisno je kod uspoređivanja proporcija između određenih kategorija prema veličini njihovih površina. Površina kategorije je zbroj svih površina svojih potkategorija. Osoba zaslužna kod kreiranja ovakvih mapa je Ben Shneiderman. Ovakvu vrstu vizualizacije podataka je kreirao kako bi prikazao veliku količinu podataka koja mu ne bi uzimala previše mjesta na zaslonu. [5]



Slika 7. Primjer mapiranja slika [20]

3. PRAVILA PRIKAZA PODATAKA S ASPEKTA DIZAJNA

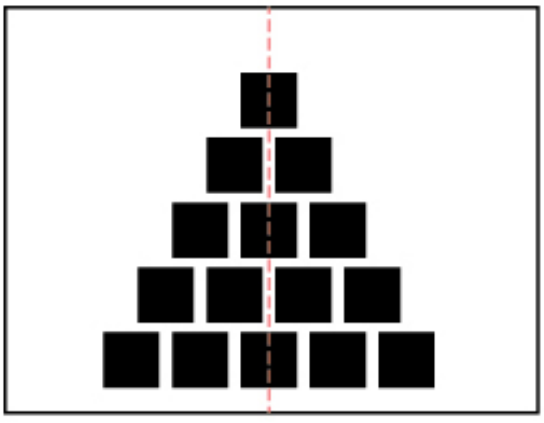
Vizualizacija podataka svoj veliki element postojanja zasniva na dizajnu. Zašto uopće nešto prikazivati, ako se ne misli potrošiti vrijeme na izgled onog što prikazujemo. Ovo su neka pravila dizajna koja su bitna kod vizualizacije podataka. [6], [7]

- Balansiranje dizajna
- Naglašavanje bitnih aspekata
- Jednostavnost
- Pametno korištenje uzoraka
- Proporcije
- Točan ritam
- Raznolikost
- Tema

3.1. Balansiranje dizajna

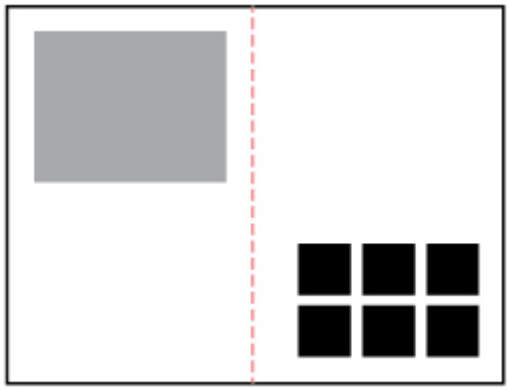
Balansirani dizajn ne mora nužno značiti simetričnost. U ovom se kontekstu odnosi na onaj dizajn koji sadrži razne vizualne elemente kao što su boje, negativan prostor ili pak teksture koje su ravnomjerno distribuirane preko ekrana. Postoje tri tipa balansa u dizajnu, a to su:

- Simetrični balans – balans u kojem je jedna strana identična drugoj



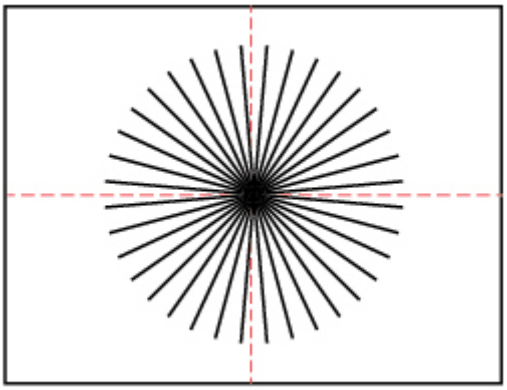
Slika 8. Simetrični balans [21]

- Asimetrični balans – balans u kojem su i jedna i druga strana različite, ali imaju podjednaku težinu



Slika 9. Asimetrični balans [21]

- Radijalni balans – balans u kojem su elementi ravnomjerno posloženi oko središnjeg objekta

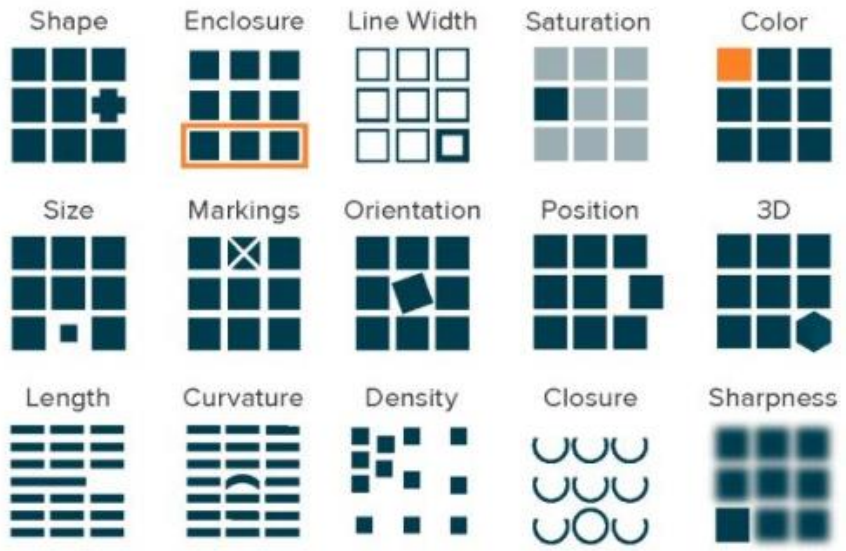


Slika 10. Radijalni balans [21]

Odgovornost je dizajnera da shvati kakav balans je potreban kod prikaza podataka. Nikada nije poželjno koristiti isti balans (klonirati) već je potrebno razmišljati izvan okvira. [6], [7]

3.2. Naglašavanje bitnih aspekata

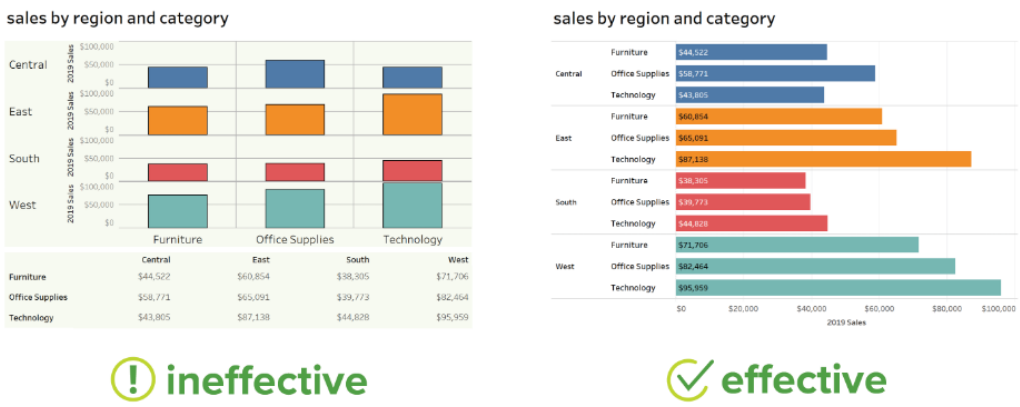
Fokus korisnika mora biti na najbitnijim dijelovima. Dizajner mora pomno odabrati pravu boju, kontrast, veličinu i negativan prostor. Najbitnije od svega je to što bitni podaci ne smiju biti nezapaženi. Bitni podaci su poželjni da budu postavljeni u gornji lijevi kut, zbog toga što je korisnikova pažnja privučena prema tom kvadrantu. [6], [7]



Slika 11. Naglašavanje bitnih aspekata

3.3. Jednostavnost

Bitno je da su vizualni elementi jednostavni i razumljivi te da nema nikakvih nepotrebnih informacija ili će u suprotnom podaci biti nečitljivi. Pretrpanost informacijama ne odgovara stvarnom cilju vizualizacije podataka. [7]

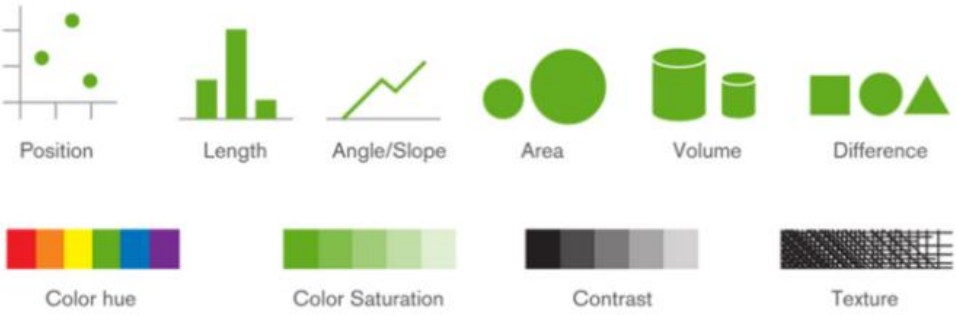


Slika 12. Jednostavnost [25]

3.4. Pametno korištenje uzoraka

Za slične informacije, poželjno je korištenje uzoraka. Uzorci se dobivaju korištenjem istih grafikona, boja, dijagrama, tekstura ili bilo kojih drugih elemenata. Korištenjem uzoraka,

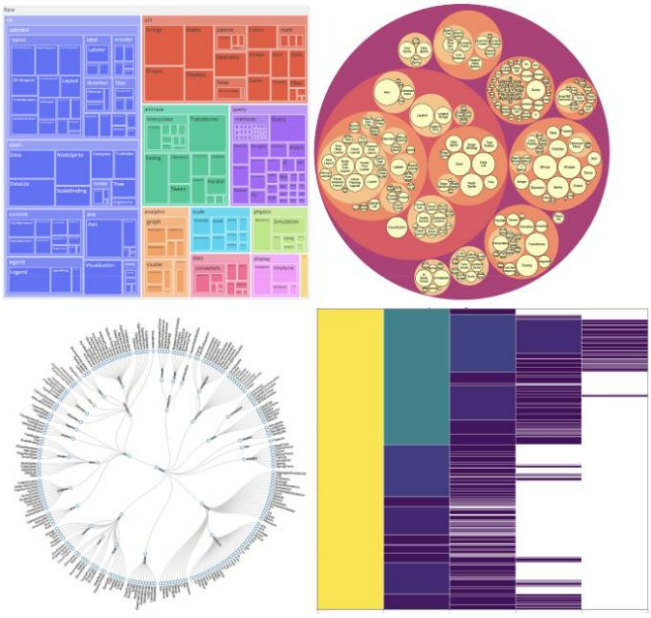
lakše se prepoznaju pogreške, odstupanja ili anomalije što je zapravo jedan od bitnijih faktora njihovog korištenja. [6], [7]



Slika 13. Pametno korištenje uzoraka [26]

3.5. Proporcije

Proporcije su bitne kod vizualizacije podataka zbog toga što indirektno uspoređuju odnose među podacima. Ukoliko želimo istaknuti neki set podataka, tj. njihove vrijednosti, moramo ih učiniti nešto većim od ostatka, naglasiti da su bitniji te da pažnja korisnika mora biti najprije usmjerena na njih. Najbolji primjer su kružni grafikoni u kojima se lako mogu vidjeti proporcije između različitih podataka. [6]

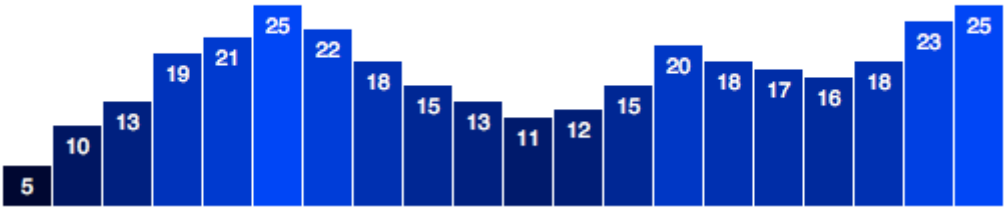


Slika 14. Proporcije [27]

3.6. Točan ritam

Ritam u ovom kontekstu predstavlja pravila pokreta i tranzicija. Za dizajn se kaže da ima točan ritam kad svi elementi dizajna koje korisnik vidi, stvaraju oku ugodan pokret ili

tranziciju. Ukoliko elementi nemaju točan ritam, to je iz razloga što pokreti nisu nesmetani, već teški, pa je teže za korisnika da se mu oči priviknu na iščitavanje podataka. [6]



Slika 15. Točan ritam [28]

3.7. Raznolikost

Možemo pretpostaviti, da će većina korisnika koji će pregledavati naše podatke imati kratku pozornost. Kako bi to popravili, naš je cilj obasuti korisnika informacijama koje će ga zanimati i održati angažiranim. Tu nastupa raznolikost. Ne samo podataka, već vizualnih elemenata koji reprezentiraju te podatke. Takvi elementi će privući poglede i pomoći korisniku da lakše zapamti informacije koje je pročitao. [6]

Slika 16. Raznolikost [29]

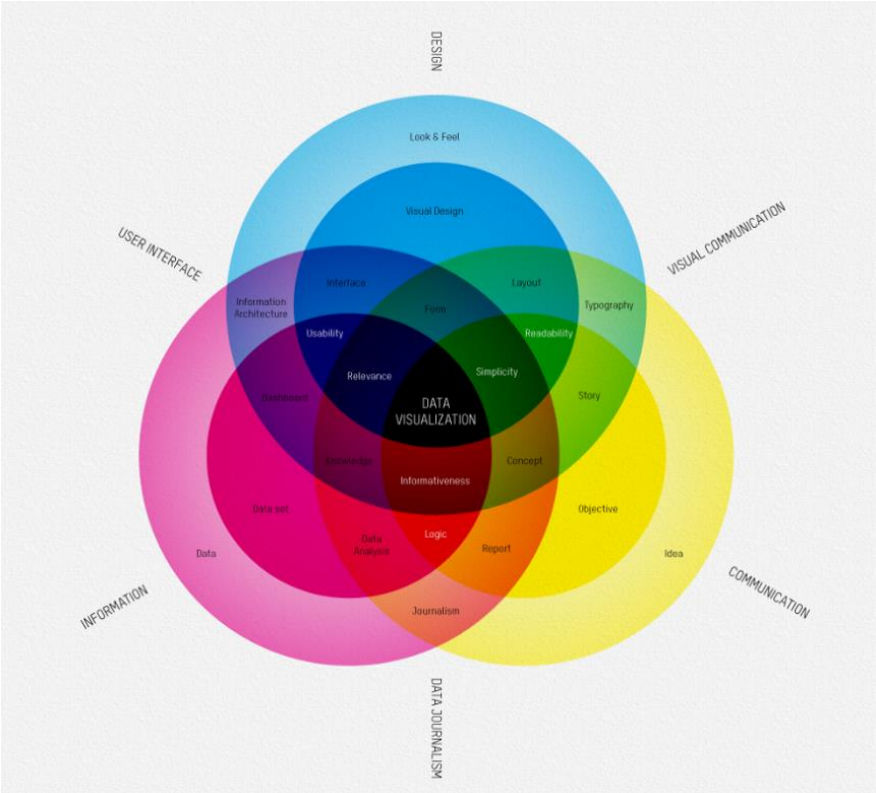
3.8. Tema

Tema predstavlja standard koji se koristi u vizualizaciji. Ovdje je bitna konzistencija dizajna. Ta se tema dobiva automatski, ako se dizajn radi na pravi način. Ukoliko korisnik vidi jedinstveni standard na svakoj stranici, bit će mu lakše razumjeti kako stranica funkcionira. [6]

4. VIZUALIZACIJA PODATAKA NA WEBU

Vizualizacija podataka može omogućiti strukturu u naizgled nestrukturiranim podacima. Web je postao medij pomoću kojeg se izrađuju razne aplikacije namijenjene upravo za vizualizaciju podataka. Mora se reći da je takvo nešto i očekivano, zbog toga što je sam web postao sinonim za izvor informacija. Vizualizacija podataka bazirana na webu omogućuje lakšu implementaciju multimedijskih sadržaja s weba u aplikacije. Web je bogat takvim aplikacijama zbog toga što je fleksibilan te je prepun informacija. [33]

Kako bi mogli napraviti aplikaciju pomoću koje prikazujemo podatke, te podatke je bitno razumjeti, potrebno je razumjeti na koji način ćemo predstaviti podatke te koji dijelovi su nešto bitniji da bi ih se stavilo u fokus. Također, potrebno je poznavanje publike koja će te podatke koristiti. Na posljetku, najbitnije je odabrati točan alat za vizualizaciju te točan vizualni prikaz koji će biti najbolja reprezentacija podataka koje prikazujemo. Vizualizacija podataka je presjek dizajna, komunikacije, vizualne komunikacije, podatkovnog novinarstva, informacija te korisničkog sučelja. Ta veza je prikazana na Vennovom dijagramu na slici 17. [34]

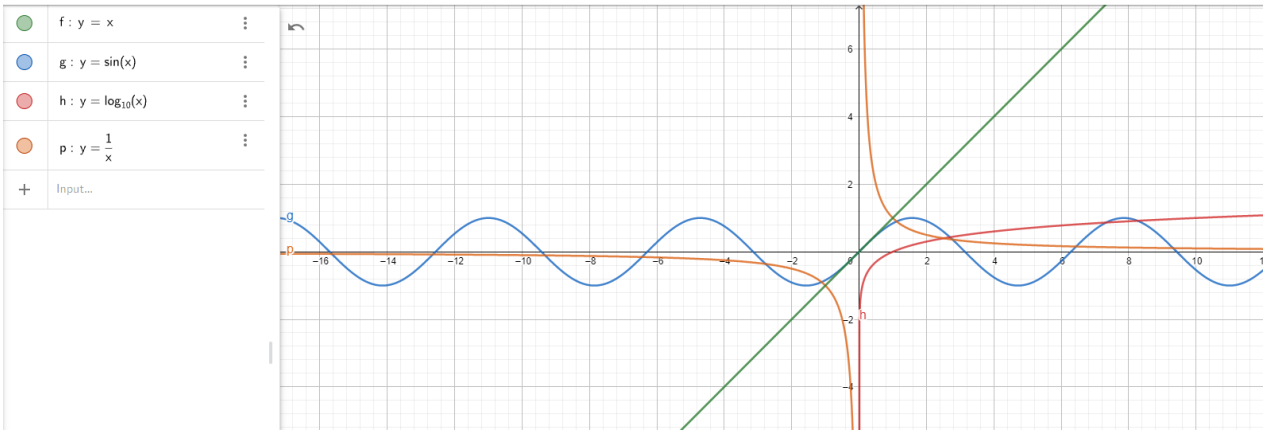


Slika 17. Vizualizacija podataka prikazana pomoću Vennovog dijagrama [34]

Na dijagramu se vidi kako svaki od elemenata ima podklasu koja na neki način reprezentira upravo taj element. Za dizajn je to izgled i osjećaj, za korisničko sučelje je arhitektura informacija, za informaciju su podaci, kod komunikacije je to ideja. Svaka od tih podklasa se međusobno presijeca, a u sredini svega toga se nalazi vizualizacija podataka.

4.1. STATIČKA I DINAMIČKA VIZUALIZACIJA

Statička vizualizacija je onaj tip vizualizacije koja se ne mijenja i koja nije u interakciji s korisnikom. Dizajn se takvom tipu vizualizacije mijenja ručno, uređivanjem samog dokumenta koji sadrži vizualne prikaze s podacima. Takav tip vizualizacije je jako limitiran i ne privlači korisnike. S druge strane, postoji i dinamički tip vizualizacije koji je sasvim suprotan statičkom, dizajn se mijenja korisnikovim pritiskom tipke na tipkovnici, prelaskom miša preko određenog elementa ili pak pritiskom na miš. Ovakav tip vizualizacije tjera korisnike da duže budu zakačeni te da istražuju razne mogućnosti koje web stranica može sadržavati. [35]



Slika 18. Dinamički graf

Na slici 18 je prikazan dinamički graf nacrtan u alatu GeoGebra u kojem se lako mogu upisati funkcije iz kojih će se generirati grafovi koji odgovaraju tim funkcijama. Ukoliko bih promijenio bilo koju od funkcija (npr. dodao 2x umjesto x), GeoGebra bi učitala novi graf.

4.2. ALATI ZA DINAMIČKU VIZUALIZACIJU

Alati za vizualizaciju općenito predstavljaju programe, web stranice, aplikacije i programske jezike koje olakšavaju reprezentaciju dobivenih podataka. Glavna svrha je "balansiranje" opterećenja prema dizajneru. Alat će prikazati podatke u onom obliku u kojem ih dizajner zada, dok će dizajner morati naučiti kako baratati samim alatom. U nastavku su prikazani najbolji alati za dinamičku vizualizaciju podataka prema [36].

- Tableau
- Infogram
- ChartBlocks

- Datawrapper
- D3.js
- Google Charts
- FusionCharts
- Chart.js
- Grafana
- Chartist.js
- Sigmajs
- Polymaps

Svaki od ovih alata ima svoje prednosti i mane. Za alat koji sam ja koristio u izradi svog rješenja, D3.js, prema [36] ima sljedeće prednosti – “snažan i prilagodljiv alat, moguća je izrada velikog broj grafikona i dijagrama, fokusiran na web standarde, postoje mogućnosti kod kojih i obični korisnici mogu kreirati vizualne prikaze (ne samo programeri), besplatan. Mane su sljedeće – Potrebno znanje izrade programa i web stranica u slučaju da se alat želi koristiti samostalno, dostupna je manja podrška nego s plaćenim alatima“. Sve je vidljivije da se specifični alati bolje snalaze u generiranju specifičnih prikaza vizualizacije, tj. dolazi do specijalizacije, pa s toga nije poželjno za više prikaza koristiti jedan alat.

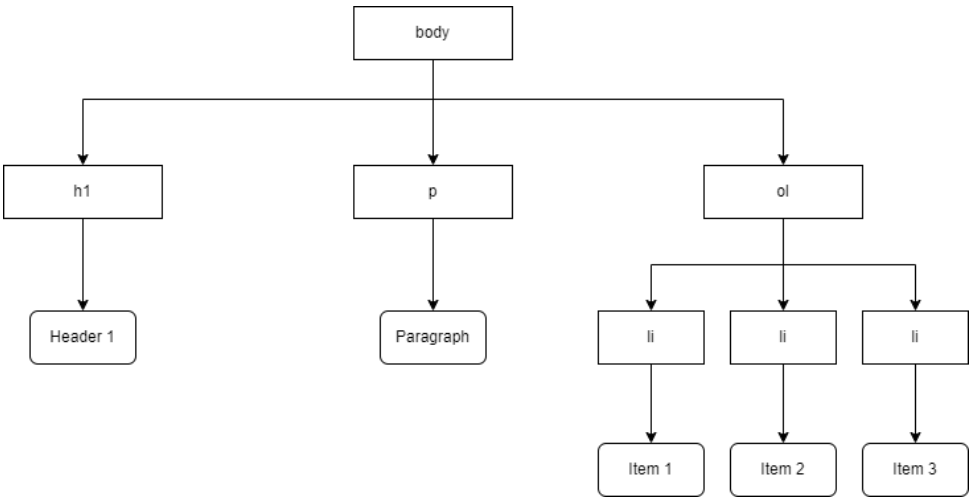
5. MANIPULACIJA DOM-OM

5.1. DOM

DOM ili *Document Object Model* je programski API (*Application Programming Interface*) za HTML i XML dokumente koji elemente nekog od tih tipova dokumenta pretvara u stablastu strukturu (engl. *tree structure*).

```
<body>
   <h1>Header 1</h1>
   <p>Paragraph</p>
   <ol>
      <li>Item 1</li>
      <li>Item 2</li>
      <li>Item 3</li>
   </ol>
</body>
```

Slika 19. Primjer HTML dokumenta



Slika 20. DOM za primjer na Slici 11.

Glavni element, tj. onaj na najvišoj razini je korijenski element (engl. *root element*). Na ovom primjeru, može se vidjeti kako taj element sadrži 3 pod-elementa ili 3 grane. Svaka od tih grana može sadržavati svoje pod-elemente. Broj razina DOM-a nije limitiran. DOM je kao

specifikacija omogućila programima u Javi i JavaScriptu da budu prijenosni kroz sve preglednike. [8]

5.2. Manipulacija

Sa DOM-om, programeri mogu kreirati dokumente, kreirati, brisati i modificirati elemente na njima, pregledavati strukturu cijelog dokumenta itd. Manipulacija predstavlja interakciju između DOM API-ja i HTML (ili XML) dokumenta tako da mu promjeni stil, unutrašnjost elementa,… Manipulacija se može dogoditi na razne načine, različitim akcijama, bilo to prelazak mišem preko elementa, pritisak na element, dolazak novih podataka (kao što će biti slučaj kod D3 biblioteke) ili postavljanje *timera*. Najbolji programski jezik za manipulaciju DOM-om je JavaScript.

U HTML-u je moguće na neki način identificirati sve elemente da bi ih mogli razlikovati jedan od drugog. Jedan od atributa koji možemo koristiti je “id“. ID predstavlja točno jedan element u cijelom dokumentu što znači da ako stavimo isti ID za 2 ili više elemenata, doći će do pogreške. U slučaju da želimo raditi s više elemenata u isto vrijeme, možemo ih klasificirati, tj. dodati atribut “class“. Dozvoljeno je da više elemenata ima istu klasu. U tom slučaju, svim tim elementima će se promijeniti stil ili funkcionalnost. Osim ID-eva i klasa, elementima se može manipulirati pomoću njihove oznake, odnosno riječi koja se nalazi između kutnih (izlomljenih) zagrada. Npr. na slici 11, oznake su: body, p, h1, ol, li. U JavaScriptu, postoji nekoliko naredbi pomoću kojih možemo dohvatiti elemente. Neke od tih naredbi su:

- getElementById
- getElementByClassName
- getElementByTagName
- querySelector
- querySelectorAll

Ove naredbe nije teško za razumjeti. “getElementById“ prima jedan parametar, *string* sa nazivom ID-a. Ista stvar je kod “getElementByClassName“ i “getElementByTagName“, samo što su u ovom slučaju to ime klase odnosno ime oznake. Ostale dvije naredbe rade nešto drugačije. Obje također primaju samo jedan parametar, a to može biti ime klase ili ID-a ili oznake. Međutim, sam program ne može razumjeti o kojem tipu selektora se radi. U slučaju da se radi o oznaci, koristit ćemo samo ime te oznake (npr. querySelector(“div“)), ako želimo dohvatit element s određenim ID-om, tada stavimo predznak “#“ (npr. querySelector(“#id-name“)). U slučaju pak da želimo dohvatiti element s određenom klasom,

koristimo predznak "." (npr. querySelector(".class-name")). "querySelector" i "querySelectorAll" su slične naredbe, ali nazad vraćaju drugačije tipove podataka. "querySelector" vraća jedan element, odnosno prvi element u dokumentu koji ima selektor jednak upisanom parametru, dok "querySelectorAll" vraća polje elemenata koji sadržavaju selektor jednak upisanom parametru. [9]

Osim dohvaćanja elemenata, DOM-om možemo manipulirati tako što kreiramo nove ili brišemo već postojeće elemente. Neke metode pomoću kojih to možemo uraditi su "createElement" ako želimo novi element. "createElement" će taj novi element postaviti na vrh dokumenta. Pomoću metoda "createTextNode" te "appendElement", moguće je dodati tekst u određeni element. Dolje se nalazi primjer. [9]

```
const newDiv = document.createElement("div"); -> kreiranje novog elementa
                                                  oznake div
const newContent = document.createTextNode("Hi there and greetings!"); -> dodavanje
                                                                  novog tekstnog sadržaja
newDiv.appendChild(newContent); -> dodjela tekstnog sadržaja
                                   novokreiranom elementu
```

Slika 21. Primjer kreiranja novog elementa s tekstom

Kod DOM-a, često se spominjalo o pojmovima kao što su elementi ili dokumenti, ali što je to zapravo? To su samo neki od fundamentalnih tipova podataka kod *Document Object* Modela. U nastavku će biti prikazani svi ostali tipovi podataka.

5.3. Document (Dokument)

Dokument predstavlja bilo koju web-stranicu učitanu u preglednik te služi kao ulazna točka u sadržaj web-stranice, tj. u *root* element stabla. Sučelje dokumenta opisuje uobičajena svojstva i metode za bilo koju vrstu dokumenta. HTML dokumenti implementiraju HTMLDocument sučelje dok SVG i XML dokumenti implementiraju XMLDocument sučelje. DOM stablo je potrebno jer ono globalno dodjeljuje funkcionalnosti dokumentu kao što su recimo kreiranje novih elemenata ili dobivanje poveznice za web-stranicu. [10]

5.4. Node (Čvor)

Svaki objekt u dokumentu je *Node*. *Node* je zapravo apstraktna klasa prema kojoj su svi ostali tipovi podataka utemeljeni. To znači da dobivaju neka svojstva *Node*-a, ali je njihova implementacija nešto složenija (nadgrađenija). U posebnim slučajevima, neke značajke *Node*

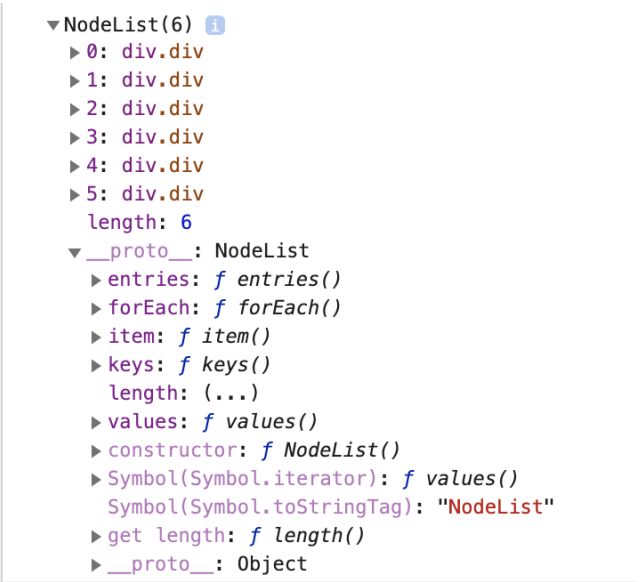
sučelja se neće primijeniti na sučelja djece, pa će se u tom slučaju vratiti *null* vrijednost ili iznimka (engl. *exception*). [10]

5.5. Element

Element je najopćenitija klasa koju nasljeđuju svi objekti koji predstavljaju elemente u dokumentu. Ona neka svoja svojstva nasljeđuje od već prije spomenutog *Node*-a. Sadrži metode i svojstva koja su zajednička svim vrstama elementa, što znači da postoje specifičniji, tj. više specijalizirani tipovi podataka. Kao što je slučaj s dokumentom, sučelje HTMLElement je baza za HTML elemente, dok je sučelje SVGElement baza za SVG elemente. [10]

5.6. NodeList

"NodeList" predstavlja polje elemenata. Jedna od već prije spomenutih metoda vraća vrijednost tog tipa, a to je "querySelectorAll". Polje je indeksirano pa se može na dva načina pristupiti određenom elementu. Polje nazvano *array* je tipa "NodeList". Prvi način pristupanja elementu je pomoću array[index_num]. Drugi način je array.item(indeks_num). Postoje dvije vrste NodeList-a. *Live* i *Static*. Live NodeList znači da će promjene u DOM-u direktno utjecati na kolekciju podataka unutar NodeList-e. S druge strane, na *Static* NodeList-e ne utječu promjene u DOM-u što znači da se kolekcija ne mijenja. Tu razliku je najbolje provjeriti sa veličinom polja. Naime, ukoliko *Live* "NodeList" naziva "parent" sadrži dva elementa, te u konzoli ispišemo parent.childNodes.length, vratit će broj 2. Ako u kolekciju dodamo novi element pomoću metode parent.appendChild(document.createElement('div')), te zatim ispišemo duljinu kolekcije u konzolu, vratit će broj 3. [10]



Slika 22. Prikaz NodeList-e u konzoli [22]

5.7. Attr

Atributi više nisu masovno korišteni, ali imaju slični definiciju kao elementi. Njihova bazična klasa koju nasljeđuju je *Node*. Sučelje “Attr“ predstavlja jedan od atributa nekog elementa. Jedina veća korist ovog tipa je što je vraćena u metodi getAttributeNode(), ali i to je nepotrebno zbog toga što metoda getAttribute() vraća mnogo jasniji naziv atributa u obliku *stringa*. Glavni cilj tipa “Attr“ je veza između imena i vrijednosti (engl. *name and value*). [10]

5.8. NamedNodeMap

“NamedNodeMap“ predstavlja polje ili kolekciju objekata tipa “Attr“. Podaci unutar takvog polja nisu poredani. Osim imenom, mogu kao i kod NodeList-e biti pristupani pomoću indeksa. “NamedNodeMap“ je *live* tip objekta što znači da će mu promjene biti automatski ažurirane, ukoliko mu se sadržaj interno promijenio. [10]

6. D3.js biblioteka

D3.js (Data-Driven Documents) je JavaScript-ova biblioteka korištena za manipulaciju DOM-om temeljem podataka. D3 je realiziran korištenjem HTML-a, CSS-a te SVG-a. Također, omogućava povezivanje bitnih podataka s DOM-om, pa samim time ga i mijenja, odnosno transformira uz mogućnost unošenja nesmetanih tranzicija. D3 je brz, fleksibilan, optimiziran za rad s velikim setovima podataka te optimiziran za rad s tranzicijama i interakcijama korisnika. [11]

Slika 23. D3 logo [23]

6.1. Selekcije

Ako bi željeli bez D3 biblioteke svim paragrafima u dokumentu dodati istu boju, trebali bi najprije dohvatiti sve elemente metodom getElementByTagName('p'), pa zatim sa for petljom promijeniti stil svakom od njih. Kod D3, sve se to može postići u jednom redu, sa selekcijom. U prijašnjem primjeru, to bi izgledalo ovako:

```
d3.selectAll("p").style("color", "blue");
```

Treba se znati da kako bi mogli uopće raditi sa D3, moramo uvesti u HTML ovaj isječak koda:

```
<script src="https://d3js.org/d3.v7.min.js"></script>
```

Ovaj isječak koda mora biti uvezen prije nego što uvezemo svoju .js datoteku. Osim "selectAll", postoji i selektor "select" koji će dohvatiti prvi element sa selektorom jednakim dodanom parametru. Po istom principu kao i "querySelector", ako želimo dohvatiti klasu i ID, moramo dodati prefiks . (točka), odnosno # (*hash*). Svaka D3 selekcija je polje čvorova. [11]

6.2. Dinamička svojstva

Dinamičko svojstvo zapravo označava da se sve vrijednosti za određena svojstva (npr. “color“, “background-color“, “font-size“,…) mogu izračunati pomoću funkcija, vraćena vrijednost funkcije će predstavljati vrijednost svojstva kojeg želimo promijeniti.

```
d3.selectAll('div')
    .data (['#000', '#fff', '#eee'])
    .style ('background-color', function(d){
            return d;
    })
```

U kodu iznad, najprije dohvaćamo sve elemente s oznakom “div“, zatim ih spajamo s poljem podataka te na kraju za svojstvo “background-color“ dobivamo vrijednost pomoću funkcije. Parametar “d“ predstavlja prije spomenute podatke. Funkcija će alternirati između boja te će svaki div element biti obojan drugom. [11]

6.3. Enter i Exit selekcije

Korištenjem selekcije *enter*, D3 stvara nove elemente kako bi odgovarali polju podataka. Svaki čvor će korespondirati jednom od elemenata u polju podataka. Ukoliko ne stavimo *enter* selekciju, a imamo više čvorova nego elemenata u polju podataka, dohvatit će se samo onoliko čvorova koliko ima elemenata u polju podataka. Za brisanje čvorova, koristi se *exit* selekcija zajedno s metodom *remove*. Preporučeno je da se *enter* i *exit* koriste odvojeno pri čemu će kod izgledati čistije i razumljivije. [11]

6.4. Tranzicije

Tranzicije su fokusirane na animirane prijelaze iz jednog stanja u drugo (to može biti stanje dijagrama, karte ili grafikona). Sve započinje s metodom *transition*. Može se dodavati trajanje animacije metodom *duration*, u zagradu se upisuje broj milisekundi. Također se može dodavati kašnjenje tranzicije koji također prima jedan parametar koji predstavlja broj milisekundi. [11]

6.5. D3 Geo

Jedna od mogućnosti D3 biblioteka koja je korištena u praktičnom dijelu ovog rada je “geoPath“. “geoPath“ kreira generator putanje te se koristi kao vrijednost atributa “d“ kod “path“ elementa za kreiranje linija. Za “geoPath“ je također bitna metoda “projection“. “Projection“ sadrži tip projekcije (geoAiry, geoAzimuthEqualArea, geoMercator, geoTimes,…) te također koordinate po kojima se crtaju linije. [12]

6.6. Osnove crtanja

Za crtanje je potrebno platno, kao platno se može koristiti “svg“ ili “canvas“ element. Crtanje znači dodavanje atributa na neki element tipa “circle“, “rect“, “line“,… Kod kruga, bitna su tri atributa: cx, cy, r. Cx predstavlja centar kruga po x koordinati, cy predstavlja centar kruga po y koordinati dok r predstavlja radijus kruga. Kod crtanja pravokutnika, prikvačimo element tipa “rect“ te su kod njega bitna 4 atributa: x, y, *width*, *height*. X i Y predstavljaju poziciju pravokutnika na platnu, *width* predstavlja širinu, a *height* predstavlja visinu pravokutnika. Crtanje linije također sadrži 4 atributa, najprije dodamo element “line“. Damo vrijednosti atributima x1, x2, y1, y2. X1 predstavlja x koordinatu za početnu točku, a X2 za završnu točku. Isti takav princip je za Y1 i Y2 sa y koordinatom. Poželjno je dodavati atribut “stroke“ kod sva tri spomenuta elementa za crtanje, te atribut “fill“ kod elemenata “circle“ i “rect“. [13]

7. PRAKTIČNI DIO RADA

Ideja praktičnog dijela rada je prikaz dinamike kod vizualizacije podataka koji su inicijalno statički. To znači da dobivamo neki set podataka koji se neće mijenjati kod svake korisnikove interakcije, već će naša stranica postići dinamiku među podacima za svaki korisnikov *input.* Kako bih to mogao prikazati, izradio sam jednostavnu web stranicu s kartom Republike Hrvatske te njenih 20 županija, ne uključujući Grad Zagreb. Sličicom sam za svaku županiju prikazao vremenske uvjete kroz 24 sata.

Za kreiranje ove web-stranice, korišteni su ovi resursi:

- VS Code
- ReactJS
- D3js
- WeatherAPI
- TopoJSON dokument

7.1. Opis postupka

Kako bi mogli prikazati vrijeme za svaku hrvatsku županiju, najprije trebamo generirati kartu. Za to se koristi TopoJSON dokument sa svim bitnim informacijama za crtanje putanje. Te informacije, koje su u ovom slučaju koordinate, pridružujemo tipu projekcije koji ćemo koristiti za crtanje. U ovom slučaju, to je geoMercator projekcija. Nakon toga smo definirali varijablu *path* koja ima definiranu putanju koju ćemo smjestiti u atribut naziva “d“ u elementu <path>. Prije toga smo definirali i <svg> element te unutar njega <g> element koji označava grupu elementa. Upravo taj element <g> sadrži 20 <path> elemenata koji će grupirano generirati cijelu kartu Hrvatske.

Slika 24. Karta Hrvatske

Nakon što smo prikazali Hrvatsku u punom sjaju, vrijeme je da prikažemo nekakve podatke na njoj. U ovom slučaju, to su podaci dobiveni s web stranice WeatherAPI koja nam daje podatke o trenutnom vremenu u svakoj županiji. Zagrebačka županija te Grad Zagreb su spojeni u jednu cjelinu. Kako bi dobili informacije za sve županije, napravljeno je polje za sve glavne gradove tih županija (ukupno se nalazi 20 gradova). Kroz polje prolazi “foreach loop“ te pomoću “axios“ modula, dobavlja podatke za svaki grad te ih jedan po jedan sprema u polje definirano pomoću “useState“ *hook*-a koja je zapravo jedan od razloga zašto je React moćan alat koji može olakšati programiranje. Naime, pomoću useState-a, developer može definirati stanja koji će sustav morati pamtiti. U našem slučaju, korišten je useState naziva [data, setData]. Data predstavlja element(objekt, polje objekata, *int*, *bool* izjavu, itd) dok setData predstavlja funkciju koja mijenja stanje tog elementa. Također, URL pomoću kojeg šaljemo HTTP GET zahtjev je oblika: http://api.weatherapi.com/v1/current.json?key=${apiKey}&q=${city}&aqi=no

API ključ je jedinstveni ključ pomoću kojeg se vidi kako je developeru dopušten pristup podacima na WeatherAPI web stranici dok ${city} predstavlja svaki glavni grad županije koji smo definirali u našem polju gradova. Također je korišten *hook* “useEffect“ koji izvodi operacije unutar sebe uvijek kad se promjeni stanje već prije određenog elementa koji se nalazi u uglatim zagradama.

```
useEffect(()=>{
     doSomething();
}, [state]);
```

Ukoliko ništa nije napisano u uglatim zagradama, “useEffect“ *hook* će se izvršiti samo jednom nakon učitavanja stranice (odnosno *renderiranja* svih elemenata na stranici). Podaci koje prikupimo će se zatim proslijediti u SVG element kao svojstvo (engl. *prop*). Kod za taj proces izgleda ovako:

```
return(
        <Svg data={data} />
);
```

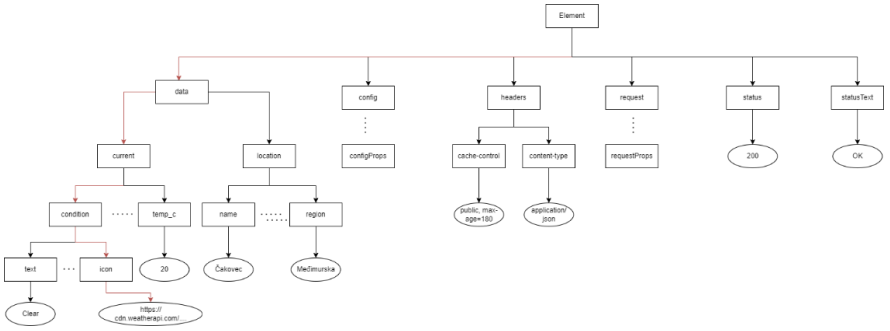
Data izvan uglatih zagrada označava naziv svojstva, dok data unutar uglatih zagrada označava podatke koje smo prikupili. Mora se napomenuti kako dobiveni podaci ne sadržavaju samo informacije o vremenu, već i druge bitne stvari kao što su temperatura (u °C i °F), brzina vjetra (u km/h i mi/h), UV indeks,… Osim toga, uključene su i informacije oko same lokacije. Prije smo spomenuli <g> element koji označava grupu te smo rekli da smo u njega stavili putanje kako bi stvorili kartu Hrvatske, ali uz <path>, postoji još jedan element. Naime, radi se o <image> elementu koji za određene vremenske uvjete, postavlja prigodnu ikonu iznad svake županije. Slike se nalaze na web-stranici našeg API-ja, dok se samom linku može pristupiti preko JSON dokumenta. Link za sliku postižemo pomoću attr() funkcije. Sama funkcija prima 2 parametra. Prvi parametar je naziv atributa, dok je drugi parametar vrijednost. U našem slučaju, vrijednost dobivamo tako što pristupamo čvorovima u JSON dokumentu. Ovo je isječak koda:

```
.attr("xlink:href", d => {
   let condition = "";
   data.forEach(el => {
     if(data.indexOf(el) !== 0 && el.data.location.name === d.properties.Capital){


       console.log(el.data.current.condition.text);
       condition = el.data.current.condition.icon;
                              }
                });
                return condition;
       })
```

Za bolje razumijevanje koda, atribut u pitanju ima naziv “xlink:href“ koji je u običnom HTML-u samo “href“. Slovo d označava geografske podatke, odnosno koordinate za svaku županiju zajedno s glavnim gradom svake županije. Glavni grad je morao biti nadodan iz razloga što ga prije nije bilo u dokumentu. Data i u ovom slučaju označava podatke o vremenu koje smo naslijedili iz prethodne komponente App.js. Mora se napomenuti, da je trenutna komponenta u kojoj se nalazimo Svg.js. Prilikom dohvata podataka, prvi element je uvijek prazan te smo stoga u uvjet stavili da se ne gleda element sa indeksom 0. Drugi dio uvjeta pronalazi identičnost između glavnih gradova županija. Naime, element geografskog JSON dokumenta

sa svojstvom “Capital“ mora odgovarati nazivu mjesta kod elementa vremenskog JSON dokumenta. Ukoliko se pronađe takav element u vremenskom JSON dokumentu, varijabli “condition“ dodjeljujemo *string* koji predstavlja potreban link. Sam *string* se nalazi u elementu vremenskog JSON dokumenta. Struktura elementa u vremenskom JSON dokumentu je prikazana na slici 25.

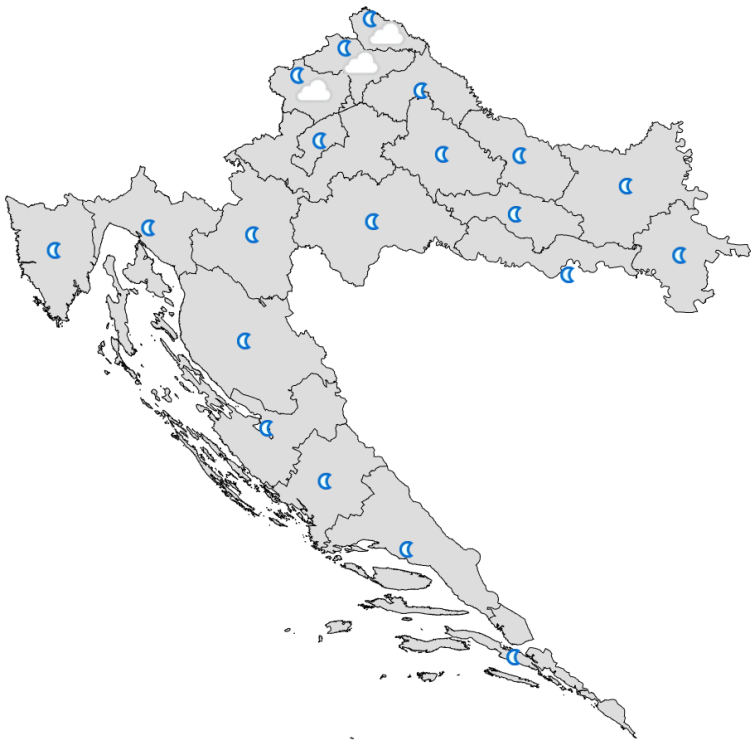


Slika 25. Prikaz jednog elementa iz vremenskog JSON dokumenta

Osim atributa “href:link“, kod slika su korišteni i “width“, “height“, “class“, “x“, “y“. X i Y su bili naročito potrebni kako bi se ikone mogle pozicionirati unutar putanja županija. Korištena je funkcija “centroid(d)[num]“. Slovo d označava geografske podatke, tj. koordinate te na temelju njih pronalazi središte putanje. “Num“ u ovom slučaju može biti 0 ili 1. 0 označava X poziciju, dok 1 označava Y poziciju. Kako središte putanje u ovom trenutku predstavlja početak slika, a ne središte slike, bitno je nadodati kako sveukupna izračunata vrijednost za središte putanje mora smanjiti za 32 piksela. To je iz razloga što su “width“ i “height“ postavljeni na 64 piksela. X atribut naposljetku izgleda ovako:

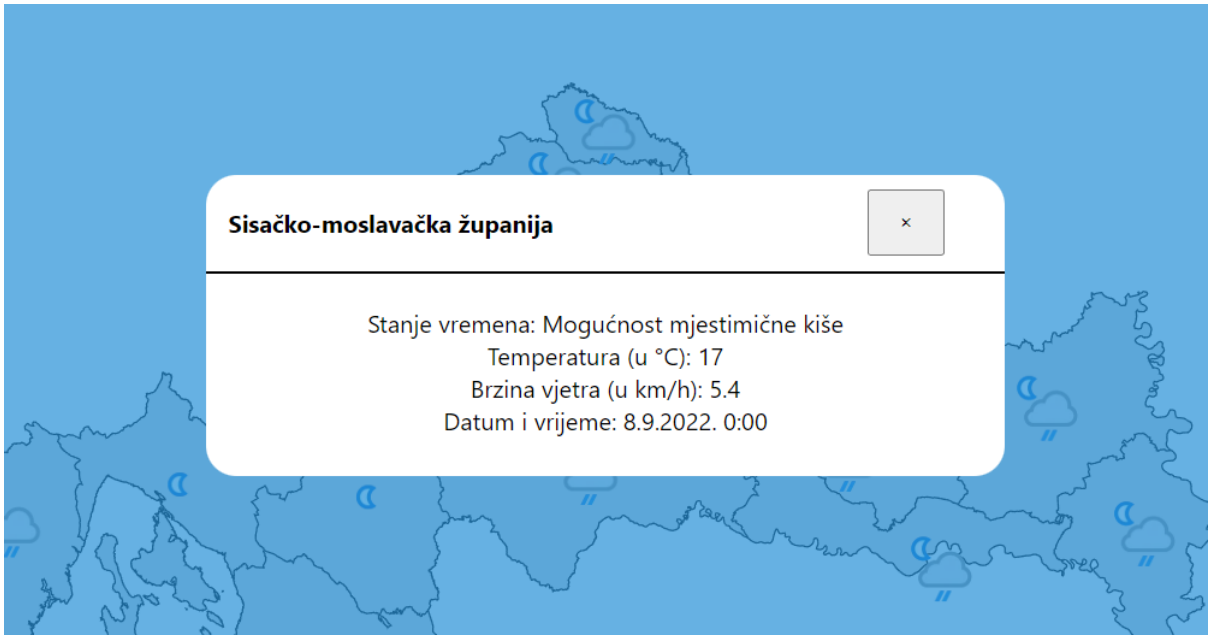
```
.attr("x", (d) => {
        return path.centroid(d)[0]-32;
})
```

Ista stvar se mora dodati i za Y koordinatu. Nakon toga, središte putanje se pomakne prema gornjem lijevom kutu gdje će se nalaziti početak ikone, stoga će i sama ikona biti točno na sredini svake županije. Karta Hrvatske nakon ovih dodanih funkcionalnosti je prikazana na slici 26.



Slika 26. Prikaz karte RH nakon dodavanja prije spomenutih funkcionalnosti

Na posljetku, kako bi se dobila dinamička vizualizacija, ugradio sam element *input* tipa “range“ s minimalnom vrijednošću 0 i maksimalnom 23 (za svaki sat u danu). Svakom promjenom *inputa* novo stanje koje smo implementirali, bit će promijenjeno. U tom stanju, nalazi se broj od 0 do 23 koji korespondira s već prije spomenutom minimalnom i maksimalnom vrijednošću. Nakon promjene stanja, React radi na način da ponovno prikaze (*re-render*) sve elemente u trenutnoj komponenti zajedno sa njenom djecom, ukoliko ih ima. Aplikacija zatim pronađe sve informacije o vremenu u satu koji je zapisan u stanju “time“ zajedno s metodom “setTime“. Pritiskom na neku od županija, također se prikazuje skočni prozor s još više informacija (brzina vjetra, temperatura,…). Prikazom skočnog prozora, ostatak aplikacije “poplavi“, a vidljivost (engl. *opacity*) bude postavljena na 71%. Naslov skočnog prozora je ime županije na koju smo pritisnuli, dok je tijelo skočnog prozora centrirano s već prije spomenutim svojstvima. Skočni prozor je prikazan na slici 27.



Slika 27. Prikaz skočnog prozora

U aplikaciji se nalazi mogućnost pokretanja animacija pritiskom na gumb "Pokreni". Osim pokretanja, animaciju je moguće zaustaviti pritiskom na gumb "Zaustavi". Pritiskom na taj gumb, indikator će se postaviti na 0:00 sati. Ukoliko se indikator trenutno nalazi na 8 sati, animacija će krenuti od 8 sati nadalje. Postoji i treći gumb, "Pauziraj" koji radi na sličan način kao i "Zaustavi", osim što taj gumb ne postavlja indikator na 0, već ga postavlja na trenutni broj sati. Ovo je postignuto sa rukovateljem događaja (engl. *event handler*). *Callback* funkcija (funkcija koja se ponaša kao argument neke druge funkcije) unutar *event handlera* postignuta je kontinuiranim pozivanjem pomoću funkcije "setInterval". Sam interval koji smo definirali, prestaje jednom kada broj sati dođe do 23. Funkcije kod gumba "Zaustavi" i "Pauziraj" rade na takav način da prekinu interval koji je u našem kodu globalna varijabla (zbog toga je možemo koristiti u sva 3 *event-handlera*). "Pauziraj" sprema trenutno stanje, dok "Zaustavi" postavlja vrijeme na 0 (uz pomoć "setTime(0)" React *hook-a*). Kod ispod prikazuje funkciju koja se pokreće pritiskom na gumb "Pokreni":

```
document.querySelector(".pokreni").addEventListener('click', () =>{
  if(time < 23){
     let i = time
     document.querySelector("#hourChange").value = i
     setTimeState()
     setBubble()
     i++
     clearInterval(timer)
     timer = setInterval(() =>{
       document.querySelector("#hourChange").value = i
```

```
        setTimeState()
        setBubble()
        console.log(time)
        i++
        if(i === 24) clearInterval(timer)
      }, 2000)
   }else{
     return
   }
})
```

Kod za gumb “Zaustavi“:

```
document.querySelector(".zaustavi").addEventListener('click', () =>{
      clearInterval(timer)
      document.querySelector("#hourChange").value = 0
      setTimeState()
      setBubble()
    })
```

Kod za gumb “Pauziraj“:

```
document.querySelector(".pauziraj").addEventListener('click', () =>{
      clearInterval(timer)
});
```

U ovim funkcijama, nalaze se dvije funkcije, jedna služi kako bi se stilski prikazao crveni “oblačić“ koji u sebi prikazuje trenutni sat, dok druga, naziva “setTimeState“ služi za brisanje trenutnih sličica iz svake županije te za promjenu vrijednosti kod *range inputa*. Kod za ove dvije funkcije se nalazi ispod:

Funkcija “setBubble()“

```
function setBubble(){
    let bubble = document.querySelector('.bubble')
    let range = document.querySelector('#hourChange')
    if(range.value < 10){
      bubble.innerHTML = '0' + range.value + ':00'
    }else{
      bubble.innerHTML = range.value + ':00'
```

```
    }
    const min = range.min ? range.min : 0;
    const max = range.max ? range.max : 100;
    const newVal = Number(((range.value - min) * 100) / (max - min))
    bubble.style.left = `calc(${newVal}% + (${8 - newVal * 0.15}px))`
}
```

Funkcija “setTimeState()“

```
function setTimeState(){
    let images = document.querySelectorAll('image')
    images.forEach(el => {
      el.remove();
    })
    let range = document.querySelector('#hourChange')
    setTime(range.value)
                                                    }
```

Funkcija “setBubble()“ najprije provjerava da li je broj sati jednoznamenkast ili dvoznamenkast. Ako je jednoznamenkast, ispred znamenke će se nadodati broj 0. Nakon toga, funkcija izračuna točnu poziciju na kojoj bi se oblačić morao nalaziti. To se ostvaruje pomoću postotaka, te se koristi formula: `((input.value - min) * 100) / (max - min))`

Funkcija “setTimeState()“ najprije briše prijašnje sličice koje reprezentiraju trenutno stanje. Nakon toga, *slider* element dobivamo pomoću funkcije document.querySelector('#hourChange'). Vrijednost tog elementa postavljamo kao trenutni sat. Sama ta promjena stanja, *renderira* komponentu u kojoj se nalazimo, te se zbog toga umeću nove sličice koje odgovaraju vremenskom stanju za taj određeni sat.

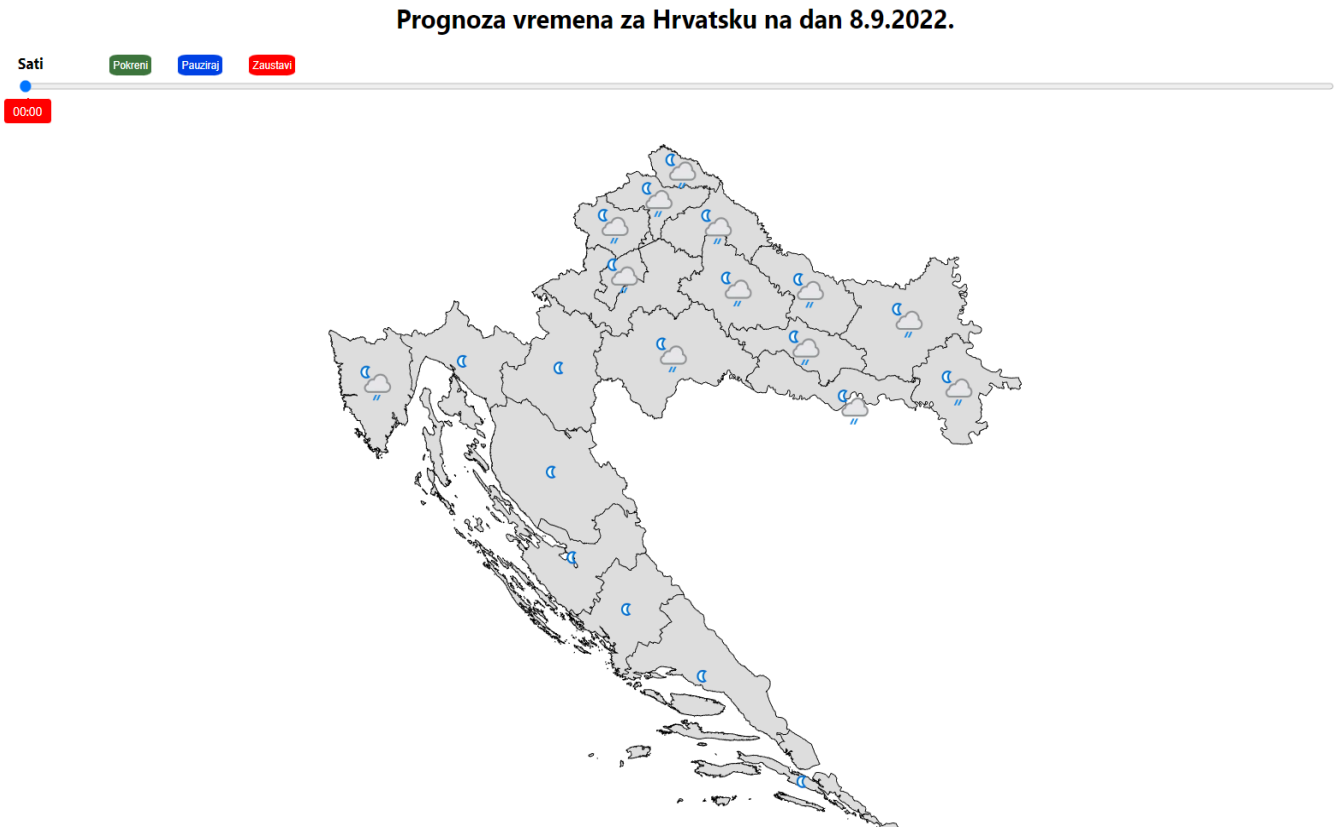
Također, kako bi skočni prozor ispisivao hrvatske nazive trenutnih vremenskih uvjeta, koja su inače napisana na engleskom, bilo je potrebno svaki od tih uvjeta napisati u zasebnom JSON dokumentu. Struktura jednog elementa u takvom dokumentu izgleda ovako:

{
"code" : 1000,
"day" : "Sunčano",
"night" : "Vedro",
"icon" : 113
}

Svojstvo “code“ predstavlja identifikator pomoću kojeg sam mogao povezati ovaj dokument s dokumentom koji mi šalje API. Svojstva “day“ i “night“ samo razlikuju nazive uvjeta koji nisu mogući (npr. za noćne vremenske uvjete se ne može staviti da je sunčano). Za taj dio, u API-ovom dokumentu je postojalo svojstvo “is_day“ tipa *bool* pomoću kojeg se vidjelo da li je za određeni sat postavljen dan ili noć.

7.2. Konačna dinamička vizualizacija

Na posljetku, web stranica izgleda kao na slici 28:



Slika 28. Prikaz gotove aplikacije

Link na aplikaciju: https://gilded-flan-93ddf7.netlify.app/

8. ZAKLJUČAK

Iz ovog rada možemo donijeti mnogo zaključaka. Jedan od tih je taj da tehnologije za vizualizaciju podataka konstantno napreduju. Vizualizaciju podataka koristimo kako bismo lakše pronalazili anomalije, odstupanja ili trendove. Zaključili smo da postoje dvije vrste vizualizacije, statička i dinamička. Dinamička vizualizacija podataka je zanimljivija korisniku zbog toga što s njom može biti u konstantnoj interakciji. Međutim, statička vizualizacija se puno brže učitava kod neke web stranice. Dokumenti takve vizualizacije mogu biti fizički, za razliku od dinamičke vizualizacije kod koje je nemoguće prikazati samu dinamiku. Postoji mnogo načina kako vizualizirati podatke, bilo to preko grafikona, dijagrama ili karata. Kako bi se olakšala kreacija tih načina vizualiziranja, možemo koristiti snažnu JavaScript-ovu biblioteku D3. D3 olakšava manipulaciju DOM-om (Document Object Model) koji sadrži sve elementa nekog HTML ili XML dokumenta u obliku stabla. Manipulacija označava modificiranje prije spomenutih elemenata, dok D3 manipulira DOM-om temeljem podataka koje mu damo. D3 je brza, fleksibilna biblioteka koja je u stanju primiti velike količine podataka, pa je iz tog razloga toliko korištena kod web developera da vizualizira te podatke. Praktični dio rada prikazao je moć vizualizacije za svakodnevno pitanje, kakvo nas vrijeme danas očekuje? Dohvaćanje pouzdanih podataka s WeatherAPI-ja u zajedništvu s raznim funkcionalnostima koje aplikacija nudi, pomoglo je da se realizira promatranje vremenskih uvjeta, temperature, pa čak i brzine vjetra u svakoj hrvatskoj županiji kroz 24 sata na dan.

9. LITERATURA

[1] Tableau (bez dat.), What Is Data Visualization? Definition, Examples, And Learning Resources [Na internetu]. Dostupno: https://www.tableau.com/learn/articles/data-visualization [pristupano 30.6.2022.]

[2] IBM (10.2.2021.), Data Visualization [Na internetu]. Dostupno: https://www.ibm.com/cloud/learn/data-visualization [pristupano 30.6.2022.]

[3] E-udžbenik (bez dat.), Izrada i oblikovanje grafikona [Na internetu]. Dostupno: https://e.udzbenik.hr/U/infOS7/72grafikoni.pdf [pristupano 30.6.2022.]

[4] CueMath (bez dat.), Line Graph [Na internetu]. Dostupno: https://www.cuemath.com/data/line-graphs/ [pristupano 30.6.2022.]

[5] The Data Visualisation Catalogue (bez dat.), Treemap [Na internetu]. Dostupno: https://www.tableau.com/learn/articles/data-visualization [pristupano 13.7.2022.]

[6] Prompt Cloud (2.3.2018.), Effective Data Visualization with Eight Design Principles [Na internetu]. Dostupno: https://www.promptcloud.com/blog/design-principles-for-effective-data-visualisation [pristupano 13.7.2022.]

[7] Medium (3.10.2019.), 7 Key Principles of Effective Data Visualization [Na internetu]. Dostupno: https://medium.com/gobeyond-ai/7-key-principles-of-effective-data-visualization-b854b0b81946 [pristupano 13.7.2022.]

[8] W3 (bez dat.), What is the Document Object Model? [Na internetu]. Dostupno: https://www.w3.org/TR/WD-DOM/introduction.html [pristupano 14.7.2022.]

[9] Medium (30.12.2020.), What Is DOM Manipulation? [Na internetu]. Dostupno: https://medium.com/swlh/what-is-dom-manipulation-dd1f701723e3 [pristupano 14.7.2022.]

[10] MDN Web Docs (bez dat.), Web APIs [Na internetu]. Dostupno: https://developer.mozilla.org/en-US/docs/Web/API [pristupano 14.7.2022.]

[11] D3 (bez dat.), Data-Driven Documents [Na internetu]. Dostupno: https://developer.mozilla.org/en-US/docs/Web/API [pristupano 15.7.2022.]

[12] D3 in Depth (bez dat.), D3 Transitions [Na internetu]. Dostupno: https://www.d3indepth.com/transitions [pristupano 15.7.2022.]

[13] D3 wiki (bez dat.), Geo Paths [Na internetu]. Dostupno: https://d3-wiki.readthedocs.io/zh_CN/master/Geo-Paths/ [pristupano 15.7.2022.]

[14] Tables as a form of data visualization [Slika] (bez dat.). Dostupno: https://learnche.org/pid/data-visualization/tables-as-a-form-of-data-visualization [pristupano 30.6.2022.]

[15] Stacked Column Chart [Slika] (bez dat.). Dostupno: https://hr.wiki-base.com/7772950-100-stacked-column-chart [pristupano 30.6.2022.]

[16] Tortni grafikon [Slika] (bez dat.) Dostupno: https://support.microsoft.com/hr-hr/office/dodavanje-tortnog-grafikona-1a5f08ae-ba40-46f2-9ed0-ff84873b7863 [pristupano 30.6.2022.]

[17] Line Graph [Slika] (bez dat.). Dostupno: https://www.cuemath.com/data/line-graphs/ [pristupano 30.6.2022.]

[18] Dobno-spolna piramida [Slika] (bez dat.). Dostupno: https://www.zagreb.hr/UserDocsImages/Demografija/Demografija.pdf [pristupano 30.6.2022]

[19] Your Friendly Weatherman's Constant [Slika] (bez dat.). Dostupno: https://vwo.com/blog/eye-tracking-heatmap/ [pristupano 30.6.2022.]

[20] ACME Products by Revenue [Slika] (bez dat.). Dostupno: https://www.anychart.com/ar/products/anychart/gallery/Tree_Map_Charts/ [pristupano 30.6.2022.]

[21] Balance in design [Slika] (bez dat.) Dostupno: https://www.quotemaster.org/balance+in+design [pristupano 13.7.2022.]

[22] NodeList [Slika] (bez dat.). Dostupno: https://dev.to/nabeelah/is-nodelist-an-array-36p3 [pristupano 14.7.2022.]

[23] D3 Logo [Slika] (bez dat.). Dostupno: https://blog.knoldus.com/introduction-to-d3-js/ [pristupano 15.7.2022.]

[24] Data Visualisation Cheat Sheet [Slika] (bez dat.). Dostupno: https://www.pinterest.com/pin/658370039254570085/ [pristupano 27.8.2022.]

[25] 6 tips for creating effective data visualizations [Slika] (bez dat.). Dostupno: https://blog.csgsolutions.com/6-tips-for-creating-effective-data-visualizations [pristupano 27.8.2022.]

[26] What are the important principles of data visualization [Slika] (19.10.2020). Dostupno: https://towardsdatascience.com/what-are-the-important-principles-of-data-visualization-3d3ca6c8c303 [pristupano: 27.8.2022]

[27] 6 Hierarchical Data Visualizations [Slika] (bez dat.). Dostupno: https://towardsdatascience.com/6-hierarchical-datavisualizations-98318851c7c5 [pristupano: 27.8.2022]

[28] Modernizing the Bar Chart [Slika] (bez dat.). Dostupno: https://www.oreilly.com/library/view/interactive-data-visualization/9781449340223/ch09.html [pristupano 27.8.2022]

[29] A detailed guide to colors in data vis style guides [Slika] (bez dat.). Dostupno: https://blog.datawrapper.de/colors-for-data-vis-style-guides/ [pristupano: 27.8.2022]

[30] Sisense (bez dat.), Charts Vs. Tables – Choosing the RightVisualization [Na internetu]. Dostupno: https://www.sisense.com/blog/charts-vs-tables-choosing-the-right-visualization/ [pristupano: 30.8.2022.]

[31] wpDataTables (9.5.2022), Charts Vs Tables or When to Use One Over the Other [Na internetu]. Dostupno: https://wpdatatables.com/charts-vs-tables/ [pristupano 30.8.2022.]

[32] hotjar (16.3.2022.), The complete guide to heatmaps. Dostupno: https://www.hotjar.com/heatmaps/ [pristupano 30.8.2022.]

[33] Rohrer R.M., Swing E. (2013.), *Web-based information visualization*. [Na internetu]. Dostupno na: https://www.researchgate.net/publication/3208623_Web-based_information_visualization [pristupano 30.8.2022.]

[34] Medium (3.6.2017.), *Data Visualization for Front-End Developers.*[Na internetu]. Dostupno na: https://medium.com/@ksykes/data-visualization-for-front-end-developers-b59953d4e13f [pristupano 30.8.2022.]

[35] Sarah Shade (2.10.2013.),*Static Vs. Dynamic Visualization* [Na internetu]. Dostupno na: https://prezi.com/roguo_f2jxa5/static-vs-dynamic-visualization/ [pristupano 30.8.2022.]

[36] Cameron Chapman (bez dat.), A Complete Overview of the Best Data Visualization Tools [Na internetu]. Dostupno na: https://www.toptal.com/designers/data-visualization/data-visualization-tools [pristupano 30.8.2022.]

10. POPIS SLIKA

Slika 1. Primjer tablice za vizualizaciju podataka……..2

Slika 2. Primjer složenog trakastog grafikona……3

Slika 3. Primjer tortnog grafikona……3

Slika 4. Primjer linijskog dijagrama……..3

Slika 5. Primjer histograma ……4

Slika 6. Primjer toplinske karte ……..4

Slika 7. Primjer mapiranja slika ……5

Slika 8. Simetrični balans……...6

Slika 9. Asimetrični balans ……7

Slika 10. Radijalni balans……...7

Slika 11. Naglašavanje bitnih aspekata……9

Slika 12. Jednostavnost……9

Slika 13. Pametno korištenje uzoraka……10

Slika 14. Proporcije……10

Slika 15. Točan ritam……11

Slika 16. Raznolikost……11

Slika 17. Vizualizacija podataka prikazana pomoću Vennovog dijagrama ……..12

Slika 18. Dinamički graf……13

Slika 19. Primjer HTML dokumenta……15

Slika 20. DOM za primjer na Slici 18.……15

Slika 21. Primjer kreiranja novog elementa s tekstom……17

Slika 22. Prikaz NodeList-e u konzoli……18

Slika 23. D3 logo……20

Slika 24. Karta Hrvatske……...23

Slika 25. Prikaz jednog elementa iz vremenskog JSON dokumenta……26

Slika 26. Prikaz karte RH nakon dodavanja prije spomenutih funkcionalnosti ……..……27

Slika 27.Prikaz skočnog prozora ……..……28

Slika 28. Prikaz gotove aplikacije……31