PANEVROPSKI UNIVERZITET APEIRON, BANJA LUKA
FAKULTET INFORMACIONIH TEHNOLOGIJA

Redovne studije
Smjer : "Inženjering Informacionih Tehnologija"

Predmet:
Viši programski jezici i RAD alati

Svelte – Pregled i implementacija u razvoju modernih web aplikacija

(Seminarski rad)

Predmetni nastavnik
Prof. dr Branimir Trenkić

Student: Pavlović Ivan
Br. Indeksa : 92-20/RITP-S

Banja Luka, Septembar 2024

SADRŽAJ

UVOD

U poslijednjih nekoliko godina, razvoj web aplikacija postao je kompleksan proces koji zahtijeva efikasne i inovativne pristupe za izgradnju brzih, responzivnih i skalabilnih rješenja. Među brojnim dostupnim tehnologijama i framework-ovima, Svelte se ističe kao revolucionaran alat u svijetu frontend razvoja. Razvijen kao alternativa tradicionalnim frameworkovima kao što su React, Vue i Angular, Svelte donosi jedinstven pristup koji se fokusira na kompilaciju koda u efikasan JavaScript kod, što rezultira poboljšanim performansama i manjim veličinama fajlova.

Ovaj rad istražuje Svelte framework, njegovu istoriju, upotrebu i komparativne prednosti u odnosu na druge popularne frameworkove. Također, razmatra strukturu Svelte komponenti, kao i način na koji se Svelte integriše sa backend tehnologijama. Analizom prednosti i mana ovog frameworka, rad pruža uvid u to kako Svelte može unaprijediti razvoj modernih web aplikacija.

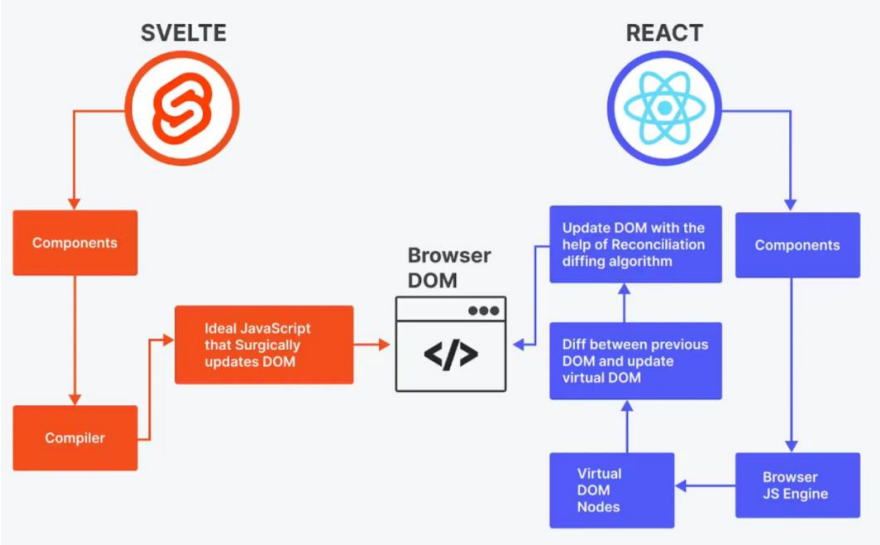
1. Svelte Framework

„Svelte je besplatan i open-source komponentno zasnovan softverski okvir za front-end razvoj, kao i jezik koji je kreirao Rich Harris i koji održavaju članovi Svelte core tima“ [1].

„Svelte nije monolitna JavaScript biblioteka koju aplikacije uvoze ali umijesto toga, Svelte kompajlira HTML šablone u specijalizovan kod koji direktno manipuliše DOM-om, što može smanjiti veličinu prenesenih fajlova i poboljšati performanse na strani klijenta. Kod aplikacije se također obrađuje od strane kompajlera, ubacujući pozive za automatsko ponovno računanje podataka i ponovni prikaz UI elemenata kada se podaci na koje se oslanjaju promjene“ [1].

Slika 1 - Svelte logo i logotip

„Ovo također izbjegava opterećenje povezano sa međureprezentacijama u runtime-u, kao što je virtuelni DOM, za razliku od tradicionalnih okvira (kao što su React i Vue) koji većinu svog posla obavljaju u runtime-u, tj. u pretraživaču“ [1].



Slika 2 - Razlika manipulacije DOM-a korištenjem Svelte Framework-a i korištenjem React Framework-a

1.1. Istorija Svelte Framework-a

„Prethodnik Svelte-a je Ractive.js, koji je Rich Harris kreirao 2013. godine.

Verzija 1 Svelte-a bila je napisana u JavaScript-u i objavljena je 29. novembra 2016. godine. U osnovi, to je bio Ractive sa kompajlerom. Ime Svelte izabrali su Rich Harris i njegovi saradnici iz The Guardian-a.

Slika 3 - Rich Harris

Verzija 2 Svelte-a objavljena je 19. aprila 2018. godine. Cilj je bio ispraviti greške koje su održavaoci vidjeli u ranijoj verziji, poput zamjene dvostrukih zagrada jednostrukim zagradama.

Verzija 3 Svelte-a bila je napisana u TypeScript-u i objavljena je 21. aprila 2019. godine. Ponovo je razmotrena reaktivnost korišćenjem kompajlera za automatsko rukovanje dodjelama promjenljivih u pozadini.

SvelteKit web okvir najavljen je u oktobru 2020. godine, a ušao je u beta fazu u martu 2021.

Verzija 4 Svelte-a objavljena je 22. juna 2023. godine. Ovo je bila održavana verzija, manja i brža od verzije 3. Dio ove verzije bila je unutrašnja prerada iz TypeScript-a nazad u JavaScript, sa JSDoc notacijama. Ova promjena izazvala je konfuziju u developerskoj zajednici, na koju je odgovorio kreator Svelte-a, Rich Harris.

Ključni rani saradnici uključuju Conduitry-ja, koji se pridružio tokom objavljivanja Svelte 1, Tan Li Hau, koji se pridružio 2019. godine, i Ben McCann, koji se pridružio 2020. godine. Rich Harris i Simon Holthausen pridružili su se kompaniji Vercel kako bi radili na Svelte-u punim radnim vremenom 2022. godine. Dominic Gannaway pridružio se Vercel-u iz React core tima kako bi radio na Svelte-u punim radnim vremenom 2023. godine“ [1].

1.2. Upotreba Svelte Framework-a

„Svelte se može koristiti za razvoj manjih dijelova interfejsa ili cijelih aplikacija. Možete započeti od nule, dozvoljavajući Svelte-u da upravlja vašim UI-jem, ili ga postepeno integrisati u postojeću aplikaciju.

Ipak, Svelte je posebno prikladan za slijedeće situacije:

- Web aplikacije za uređaje sa niskom potrošnjom: Aplikacije razvijene u Svelte-u imaju manje pakete, što je idealno za uređaje sa sporim mrežnim vezama i ograničenom procesorskom snagom. Manje koda znači manje kilobajta za preuzimanje, parsiranje, izvršavanje i zadržavanje u memoriji.

- Visoko interaktivne stranice ili složene vizualizacije: Ako pravite vizualizacije podataka koje moraju da prikažu veliki broj DOM elemenata, performanse koje donosi okvir bez opterećenja u runtime-u osiguravaju brze i odzivne korisničke interakcije.

- Uvođenje ljudi sa osnovnim znanjem o web razvoju: Svelte ima plitku krivu učenja. Web developeri sa osnovnim znanjem HTML-a, CSS-a i JavaScript-a mogu lako da savladaju Svelte-ove specifičnosti u kratkom roku i počnu da prave web aplikacije.

- Progresivne web aplikacije (PWA): Zbog svoje male veličine paketa i brzine, Svelte je odličan za kreiranje progresivnih web aplikacija. Ove aplikacije mogu raditi offline, brže se učitavati i pružiti bolje korisničko iskustvo na mobilnim uređajima.

- Server-side rendering (SSR): SvelteKit podržava SSR, omogućavajući brže prikazivanje stranica i bolju SEO optimizaciju. Ovo je korisno za aplikacije koje zahtijevaju brže inicijalne odgovore, kao što su blogovi ili e-commerce sajtovi.

- IoT kontrolne ploče: Zbog svoje niske potrošnje resursa i performansi, Svelte se može koristiti za razvoj kontrolnih ploča za IoT uređaje, gdje je važno brzo osvježavanje podataka i nisko opterećenje sistema.

- Animacije i tranzicije: Svelte ima ugrađenu podršku za animacije i tranzicije, što ga čini dobrim izborom za aplikacije sa složenim animacijama, kao što su interaktivne prezentacije ili multimedijalne platforme.

Tim koji razvija Svelte je lansirao SvelteKit, okvir za izgradnju web aplikacija koristeći Svelte. Sadrži funkcionalnosti prisutne u modernim web okvirima, kao što su rutiranje bazirano na fajl sistemu, renderovanje na strani servera (SSR), načini renderovanja specifični za stranice, podrška za offline rad i još mnogo toga“ [2].

1.3. Svelte vs. React, Vue, Angular Framework-ova

„Kada se uporedi sa drugim poznatim framework-ovima za web razvoj, kao što su React, Angular i Vue, Svelte se ističe svojim jedinstvenim prednostima. Iako svaki od ovih framework-ova ima svoje zasluge i specifične slučajeve upotrebe, Svelte se izdvaja time što je kompajler, a ne tradicionalni framework zasnovan na biblioteci.

Slika 4 - Ranging Framework-a tokom proteklih par godina i njihove upotrebe

React je poznat po velikoj podršci zajednice i bogatom izboru biblioteka. Angular se cijeni zbog stabilnosti i moćnih alatki za razvoj. Vue pruža uravnoteženu snagu u raznim aspektima web razvoja. Međutim, nijedan od ovih framework-ova ne koristi operaciju u vrijeme kompajliranja, što je srž Svelte-ove funkcionalnosti.

Posebno, u poređenju performansi, aplikacije izgrađene sa Svelte-om često rade brže od onih kreiranih sa React-om ili Angular-om. Ova prednost u brzini može se pripisati njegovoj laganoj prirodi — Svelte ne dodaje nepotrebnu težinu učitavanjem cijele biblioteke u pretraživač.

Još jedan aspekt u kojem Svelte briljira je čitljivost i smanjenje koda — njegov dizajn omogućava programerima da pišu manje linija koda, bez žrtvovanja funkcionalnosti ili snage. To dovodi do čišćih baza koda koje su lakše za održavanje i otklanjanje grešaka.” [3]

2. Svelte komponente i sintaksa

2.1. Struktura Svelte komponenti

“Svelte komponente su osnovne jedinice za izgradnju aplikacija i mogu se vidjeti kao kombinacija HTML, CSS i JavaScript u jednom fajlu. Svaka Svelte komponenta se obično nalazi u fajlu sa ekstenzijom .svelte. Struktura komponente se sastoji od tri glavna dela:

HTML - Sadrži strukturu korisničkog interfejsa (UI). Ovo je dio u kojem se definišu elementi koje će korisnici vidjeti.

```
<h1>Hello, {name}!</h1>
```

CSS - Definiše stilove koji se primjenjuju samo na ovu komponentu. Stilovi su lokalni za komponentu, što znači da ne utiču na druge komponente ili globalne stilove.

```
<style>
  h1 {
    color: blue;
  }
```

```
</style>
```

JavaScript - Služi za definisanje logike komponente, kao što su promenljive, funkcije i importovanje drugih modula. Ovdje se također definišu "reactive" promjenljive koje automatski ažuriraju UI kada se njihova vrijednost promjeni.” [4]

```
<script>
  export let name = 'World';
</script>
```

2.2. Bindovanje podataka, događaji i stilizacija

„U Svelte-u, bindovanje omogućava direktnu povezanost između UI elemenata i promjenljivih. Kada se promeni vrednost promenljive, automatski se ažurira UI i obrnuto.

```
<input bind:value={name}>
<p>Your name is: {name}</p>
```

Svelte koristi direktivu on:event za upravljanje događajima. Ovo omogućava reagovanje na različite korisničke akcije, kao što su klikovi, promene u input poljima itd.

```
<button on:click={handleClick}>Click me</button>
<script>
  function handleClick() {
    alert('Button clicked!');
  }
</script>
```

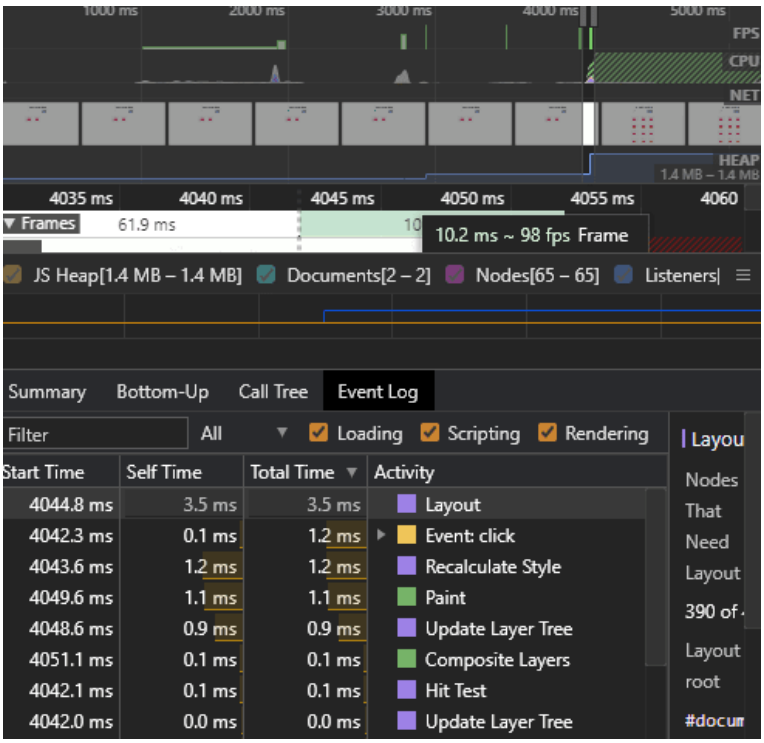
Stilizacija u Svelte-u je jednostavna i lokalizovana. Možemo koristiti <style> tag unutar komponente da definišemo stilove koji će se primjeniti samo na elemente unutar te komponente. Također možemo koristiti dinamičke stilove i klase.“ [4]

```
<style>
  .highlight {
    background-color: yellow;
  }
</style>
<p class:highlight={isHighlighted}>This is a highlighted paragraph.</p>
<script>
  let isHighlighted = true;
</script>
```

3. Prednosti i mane Svelte Framework-a

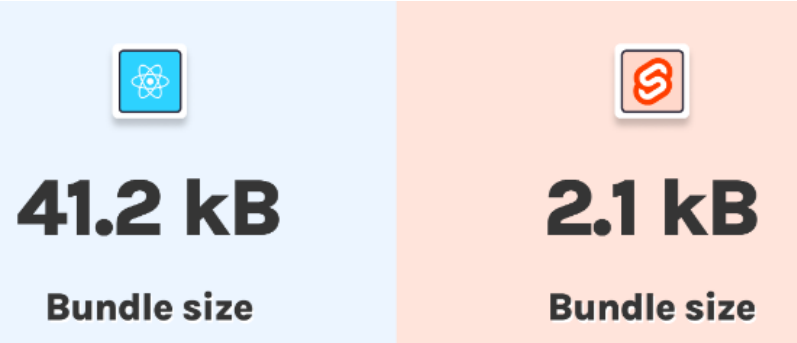
3.1. Prednosti Svelte-a

Performanse Svelte Framework-a nudi visoke performanse zbog svog kompajlerskog pristupa. Komponente se kompajliraju u optimizovani JavaScript kod koji direktno manipuliše DOM-om, što smanjuje potrebu za virtualnim DOM-om i omogućava brže ažuriranje interfejsa.



Slika 5 - Test Svelte performansi u Devtools Chrome pretraživaču

Svelte generiše manji izlazni kod u poređenju sa drugim frameworkovima. Ovo je zbog toga što se samo potrebne funkcionalnosti kompajliraju u krajnji kod, što rezultira manjim fajlovima i bržim učitavanjem stranica.



Slika 6 - Razlika u veličini datoteka između Svelte i React framework-a

Sintaksa Sveltea je jednostavna i intuitivna. Kombinovanje HTML, CSS i JavaScript u jednom fajlu čini kod lakšim za razumijevanje i održavanje. Također, Svelte ne koristi mnogo složenih koncepata, što može olakšati učenje i upotrebu.

```
<script>
  let items = [
    { id: 1, name: "Milk", done: false },
    { id: 2, name: "Bread", done: true },
    { id: 3, name: "Eggs", done: false }
  ];

  const remove = item => {
    items = items.filter(i => i !== item);
  };
</script>

<div>
  <h1>Things to Buy</h1>

  <ul>
    {#each items as item}
      <li>
        <span>{item.name}</span>
        <button on:click={() => remove(item)}>❌</button>
      </li>
    {/each}
  </ul>
</div>
```

Slika 7 - Primjer sintakse Svelte-a

3.2. Mane Svelte-a

Svelte je relativno noviji u poređenju sa frameworkovima poput React-a i Vue-a, što znači da ima manji ekosistem biblioteka i alata. Može biti teže pronaći specifične pakete ili dodatke koji su dostupni za druge frameworkove.

Iako kompajlerski pristup ima prednosti u performansama, može dodati dodatni sloj kompleksnosti u procesu razvoja. Morate upravljati build alatima i razumijevati kako Svelte kompajlira kod.

Budući da je Svelte noviji i manje rasprostranjen, može biti izazov pronaći resurse i podršku u zajednici u poređenju sa većim frameworkovima. To može uticati na brzinu rješavanja problema i dostupnost tutorijala.

U nekim slučajevima, integracija sa postojećim projektima ili alatima može biti otežana zbog specifičnih potreba Sveltea. Na primjer, prelazak sa postojećeg frameworka na Svelte može zahtijevati značajne izmjene u kodnoj bazi.

S obzirom na to da je Svelte relativno nov, postoji neizvjesnost oko dugoročne podrške i usvajanja. Iako trenutno ima podršku i koristi se u proizvodnji, budućnost može donijeti promjene u njegovom razvoju i usvajanju.

4. Integracija Svelte Framework-a sa Backend-om

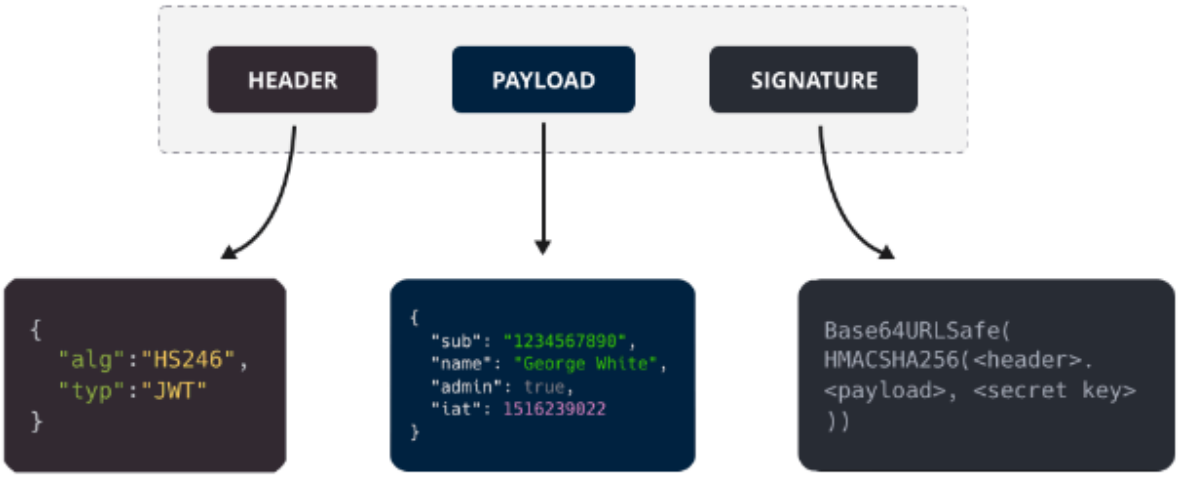
4.1. API komunikacija

Svelte može komunicirati sa backend serverima koristeći standardne JavaScript metode kao što su fetch ili Axios. API pozivi mogu se koristiti za dobijanje podataka, slanje podataka ili upravljanje stanjem aplikacije.

```
<script>
  import { onMount } from 'svelte';

  let items = [];

  onMount(async () => {
    const response = await fetch('https://api.example.com/items');
    items = await response.json();
  });
</script>

<ul>
  {#each items as item}
    <li>{item.name}</li>
  {/each}
</ul>
```

4.2. Autentifikacija i autorizacija

Za upravljanje korisnicima, možete koristiti različite metode kao što su JWT (JSON Web Tokens) ili OAuth. Na backend strani, implementirajte autentifikaciju i autorizaciju prema vašim potrebama, dok Svelte može koristiti biblioteku za rad sa sesijama i prijavama korisnika.



Slika 8 - Struktura JSON Web Token-a

4.3. WebSocket

Ako je potrebno uspostaviti real-time komunikaciju, možemo koristiti WebSocket za dvosmjernu komunikaciju između klijenta i servera.

Svelte, kao moderni framework za razvoj web aplikacija, omogućava efikasnu izgradnju korisničkog interfejsa putem kompilacije HTML, CSS i JavaScript kodova direktno u optimizovan kod koji manipuliše DOM-om. Kada se radi o real-time komunikaciji, Svelte se može efikasno kombinovati sa WebSocket-om za omogućavanje dvosmjerne komunikacije između klijenta i servera.

WebSocket je protokol koji omogućava stalnu, dvosmernu komunikaciju između klijenta i servera preko jedne TCP veze. Ovaj protokol omogućava serveru da šalje podatke klijentu u stvarnom vremenu bez potrebe da klijent konstantno šalje zahtijeve serveru (polling). Ovo čini WebSocket idealnim za aplikacije koje zahtevaju brzu i kontinuiranu razmjenu podataka, kao što su chat aplikacije, online igre ili finansijske aplikacije.

```
<script>
  let messages = [];
  const socket = new WebSocket('ws://example.com/socket');

  socket.onmessage = (event) => {
    messages = [...messages, event.data];
  };

  function sendMessage(message) {
    socket.send(message);
  }
</script>

<input bind:value={newMessage} placeholder="Type a message">
<button on:click={() => sendMessage(newMessage)}>Send</button>

<ul>
  {#each messages as message}
    <li>{message}</li>
  {/each}
</ul>
```

Svelte-ov reaktivni sistem omogućava automatsko ažuriranje korisničkog interfejsa kada se promjene podaci. U slučaju WebSocket-a, poruke koje stignu od servera mogu se dinamički dodavati u listu poruka koja se prikazuje korisniku:

```
<input bind:value={newMessage} placeholder="Type a message">
<button on:click={() => sendMessage(newMessage)}>Send</button>

<ul>
  {#each messages as message}
    <li>{message}</li>
  {/each}
</ul>
```

Važno je obezbijediti da se veza sa WebSocket serverom pravilno upravlja, uključujući otvaranje i zatvaranje veze. Ovo se može raditi u životnom ciklusu komponente Svelte:

```
<script>
  let messages = [];
  const socket = new WebSocket('ws://example.com/socket');

  socket.onmessage = (event) => {
    messages = [...messages, event.data];
  };

  function sendMessage(message) {
    socket.send(message);
  }

  // Cleanup on component destroy
  onDestroy(() => {
    socket.close();
  });
</script>
```

ZAKLJUČAK

Svelte predstavlja značajan korak naprijed u razvoju frontend tehnologija, nudeći jedinstven pristup koji se razlikuje od tradicionalnih frameworkova kao što su React, Vue i Angular. Sa svojom sposobnošću da generiše visokokvalitetan, optimizovan JavaScript kod, Svelte omogućava brže učitavanje stranica i bolje performanse, dok njegova jednostavna sintaksa i manjak boilerplate koda olakšavaju razvoj i održavanje aplikacija.

Upotreba Svelte frameworka može donijeti značajne prednosti, posebno u projektima gde su performanse i veličina fajlova ključni faktori. Ipak, postoji i nekoliko izazova, uključujući manji ekosistem i složenost u integraciji sa postojećim alatima i tehnologijama. U poređenju sa Reactom, Vueom i Angularom, Svelte nudi konkurentske prednosti, ali i određene nedostatke koji treba da se uzmu u obzir prilikom donošenja odluke o izboru tehnologije za razvoj web aplikacija.

POPIS SLIKA

CITATNI IZVORI

[1] »WikiPedia,« [Mrežno]. Available: https://en.wikipedia.org/wiki/Svelte.

[2] »MDN web docs,« [Mrežno]. Available: https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Svelte_getting_started.

[3] »Svelte.dev,« [Mrežno]. Available: https://svelte.dev/docs/basic-markup.

[4] »DEV,« [Mrežno]. Available: https://dev.to/ideoagency/svelte-the-new-frontend-framework-4ghf.