

Vue.js 3

Kuvar

Otkrijte efikasna rešenja za izradu modernih veb aplikacija pomoću najnovijih Vue funkcija i TypeScripta

Heitor Ramon Ribeiro

Vue.js 3

Otkrijte efikasna rešenja za izradu modernih veb aplikacija pomoću najnovijih Vue funkcija i TypeScripta

Heitor Ramon Ribeiro

Izdavač:



**kompjuter
biblioteka**

Obalskih radnika 4a, Beograd

Tel: 011/2520272

e-mail: kombib@gmail.com

internet: www.kombib.rs

Urednik: Mihailo J. Šolajić

Za izdavača, direktor:

Mihailo J. Šolajić

Autor: Heitor Ramon Ribeiro

Prevod: Biljana Tešić

Lektura: Miloš Jevtović

Slog: Zvonko Aleksić

Znak Kompjuter biblioteke:

Miloš Milosavljević

Štampa: „Pekograf“, Zemun

Tiraž: 500

Godina izdanja: 2021.

Broj knjige: 537

Izdanje: Prvo

ISBN: 978-86-7310-5xx-x

Vue.js 3 Cookbook

Heitor Ramon Ribeiro

ISBN 978-1-83882-622-2

Copyright © September 2020 Packt Publishing

All right reserved. No part of this book may be reproduced or transmitted in any form or by means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Autorizovani prevod sa engleskog jezika edicije u izdanju „Packt Publishing“, Copyright © September 2020.

Sva prava zadržana. Nije dozvoljeno da nijedan deo ove knjige bude reprodukovano ili snimljeno na bilo koji način ili bilo kojim sredstvom, elektronskim ili mehaničkim, uključujući fotokopiranje, snimanje ili drugi sistem presnimavanja informacija, bez dozvole izdavača.

Zaštitni znaci

Kompjuter Biblioteka i „Packt Publishing“ su pokušali da u ovoj knjizi razgraniče sve zaštitne oznake od opisnih termina, prateći stil isticanja oznaka velikim slovima.

Autor i izdavač su učinili velike napore u pripremi ove knjige, čiji je sadržaj zasnovan na poslednjem (dostupnom) izdanju softvera. Delovi rukopisa su možda zasnovani na predizdanju softvera dobijenog od strane proizvođača. Autor i izdavač ne daju nikakve garancije u pogledu kompletnosti ili tačnosti navoda iz ove knjige, niti prihvataju ikakvu odgovornost za performanse ili gubitke, odnosno oštećenja nastala kao direktna ili indirektna posledica korišćenja informacija iz ove knjige.

O AUTORU

Heitor Ramon Ribeiro razvija veb aplikacije više od 15 godina i neprestano prelazi sa frontend razvoja na backend razvoj. Prateći svoju strast prema UX/UI i programiranju, odlučio je da se bavi frontend razvojem.

Izradio je poslovne aplikacije za preduzeća koja koriste Vue.js i principe čiste arhitekture, tako što je preusmerio svoj „put“ sa zastarelih aplikacija na novi „svet“ aplikacija pomoću jednostraničnih aplikacija (Single-Page Applications – SPA) i progresivnih veb aplikacija (Progressive Web Applications – PWA). Smatra da je u današnje vreme pomoću pregledača skoro sve moguće, a da je JavaScript „budućnost“ programiranja.

Kada ne programira ili kada ne vodi frontend tim, Heitor Ramon Ribeiro se zabavlja sa porodicom, strimuje svoje sesije o igrama ili sam igra neke First Person Shooter igre.

Zahvaljujem se svojoj divnoj supruzi Raquel što mi je svakodnevno pomagala i podržavala tokom procesa objavljivanja moje prve knjige, mom sinu Marku, kojeg mnogo volim i činim sve za njega, mojoj porodici i prijateljima koji su mi pomogli tokom razvoja ove knjige, posebno Patricku Monteiru, koji mi je mnogo pomogao.

Zahvaljujem se svim saradnicima i kompanijama sa kojima sam radio poslednjih nekoliko godina, a koji su mi pomogli i shvatili koliko mi znači ovaj projekat.

O RECENZENTU

Swanand Kadam je tvorac prvog hindi programskog jezika na Internetu, koji je osmišljen da poboljša pisanje kodiranja u Indiji. On je iskusni arhitekta veb aplikacija i PWA arhitekta koji je dizajnirao i razvio e-trgovinu, upravljanje zaposlenima i prilagođena softverska rešenja za veliki broj preduzeća. Dosledni je korisnik tehnologija, uključujući Vue.js, Firebase, Node.js, Google Cloud i NoSQL baze podataka. Njegovi članci objavljeni su na sajtovima o vrhunskoj tehnologiji, kao što su „Better Programming“ i „Hackernoon“. Takođe je urednik sajta „InfoQ“, na kojem piše o najnovijim trendovima u svetu razvoja softvera.

Tyler Van Blargan je programer, koji prvenstveno koristi Vue.js i pomaže drugima da nauče veb razvoj. Kada ne razvija nove aplikacije, fotografiše svoje ljubimce i igra društvene igre.

„PACKT“ TRAŽI AUTORE KAO ŠTO STE VI

Ako ste zainteresovani da postanete autor za „Packt“, prijavite se na stranicu authors.packtpub.com. Saradujemo sa hiljadama programera i tehničkih profesionalaca da bismo im pomogli da podele svoje mišljenje sa globalnom tehničkom zajednicom. Možete da podnesete osnovnu prijavu, da se prijavite za specifičnu temu za koju tražimo autore ili da pošaljete neke svoje ideje.

Predgovor

Vue je minimalni frontend radni okvir koji omogućava programerima da kreiraju veb aplikacije, prototipove, velike poslovne aplikacije, desktop aplikacije i mobilne aplikacije.

Vue 3 je kompletno ponovo napisan Vue i sadrži izmene u svim osnovnim API-ima radnih okvira. Ovo ponovno pisanje menja kod koji je napisan u TypeScriptu. U Vueu 3 su izloženi svi osnovni API-i, pri čemu je svima omogućeno korišćenje radnog okvira Vue.

Na početku knjige ćete videti recepte za implementaciju Vue.js 3 novih funkcija u vaše projekte veb razvoja i za migraciju postojećih Vue aplikacija na najnoviju verziju. Postavićete i pokrenućete TypeScript pomoću radnog okvira Vue i pronaći ćete sažeta rešenja za uobičajene izazove i „zamke“ sa kojima se suočavate prilikom implementacije komponentata, derivata i animacija, izrade pluginova, dodavanja upravljanja stanjem, usmeravanja i razvoja kompletnih jednostraničnih aplikacija (SPA - Single-Page Applications).

Neke biblioteke, pluginovi i radni okviri korišćeni u ovoj knjizi možda će biti ažurirani dok budem pisao ovu knjigu ili dok je budete čitali. Dakle, obratite pažnju na sve promene API-a ili na verzije koje mogu imati neke najnovije izmene.

Kome je namenjena ova knjiga

Ova knjiga je namenjena veb programerima koji žele da saznaju više o radnom okviru Vue i žele da poboljšaju svoje Vue veštine. Prvo će biti predstavljene tehnologije Vue 3 i TypeScript. U sledećim poglavljima čitaocu će biti predstavljeni novi koncepti u Vueu i njegovi pluginovi za ekosisteme, UI radni okviri i napredni recepti.

Kada u celosti pročitate ovu knjigu, moći ćete da kreirate Vue aplikaciju, da koristite sve osnovne Vue pluginove i da upotrebite vrhunske Vue UI radne okvire. Ako već poznajete Vue, otkrićete nove važne obrasce.

Šta obuhvata knjiga Vue.js 3 kuvar

U Poglavlju 1, „Razumevanje Vuea 3 i kreiranje komponenata“, čitaocu su obezbeđeni recepti za upotrebu novih Vue 3 API-a za kreiranje prilagođenih Vue komponenata pomoću Vue izloženih osnovnog API-a i Composition API-a. Ovo poglavlje takođe je od pomoći čitaocu na početnom „putu“ nadgradnje aplikacije Vue 2 na Vue 3.

U Poglavlju 2, „Uvod u TypeScript i Vue Ecosystem“, upoznaćete TypeScript nadskup i kako se koristi, počev od osnovnih tipova, interfejsa i anotacija o tipovima. Bićete spremni za razvoj Vue aplikacije pomoću Vue CLI-a, TypeScripta i `vue-class-component`.

U Poglavlju 3, „Vezivanje podataka, validacija obrazaca, događaji i izračunata svojstva“, razmatrani su osnovni razvoj Vuea i koncepti komponenata, uključujući `v-model`, oslušivače događaja, izračunata svojstva i petlje `for`. Čitalac će upoznati plugin `Vuelidate` za validaciju obrazaca i kako se on koristi u Vue komponenti, zajedno sa debugovanjem u komponenti Vue pomoću `vue-devtools`.

Poglavlje 4, „Komponente, miksin i funkcionalne komponente“, vodi čitaoca kroz izradu komponenata pomoću različitih pristupa, uključujući prilagođene slotove za sadržaj, potvrđena svojstva, funkcionalne komponente i kreiranje miksinu za ponovnu upotrebu koda. Zatim će čitalac upoznati skup različitih rešenja za pristup podacima podređenih komponenata, kreiranje komponente injektovanja zavisnosti i dinamički injektovanu komponentu i način na koji se odloženo učitava komponenta.

U Poglavlju 5, „Preuzimanje podataka sa Veba pomoću HTTP zahteva“, ukazujemo čitaocu kako da kreira prilagođeni omotač oko Fetch API-a za HTTP pozive u JavaScriptu, kako da koristi omotač u Vueu i kako da implementira prilagođene asinhronne funkcije u Vueu. Čitalac će takođe naučiti kako da zameni Fetch API u omotaču za aksiome i kako da prilagođene hendlere implementira na aksiome.

U Poglavlju 6, „Upravljanje rutama pomoću `vue-router`“, prikazani su Vue plugin za rutiranje i način kako se on koristi u Vueu za kreiranje ruta za stranice Vue aplikacije. Takođe je prikazan uvod u proces upravljanja putanjama rutera, dinamička putanja sa parametrima na putanji rutera, odloženo učitanje komponenata stranice, kreiranje posredničkog softvera za autentifikaciju na ruteru i korišćenje alijasa i preusmeravanje.

U Poglavlju 7, „Upravljanje stanjem aplikacije pomoću Vuexa“, istražujemo plugin za upravljanje Vue stanjem da bi čitalac mogao da razume kako Vuex funkcioniše i kako se može primeniti na njegovu aplikaciju. Ovo poglavlje takođe obezbeđuje čitaocu recepte za kreiranje Vuex modula, radnje, mutacije i gettere i način za definisanje osnovnog stanja za skladištenje.

U Poglavlju 8, „Animiranje aplikacije pomoću tranzicija i CSS-a“, istražujemo osnovne CSS animacije i tranzicije, tako što obezbeđujemo recepte za prilagođene animacije zasnovane samo na CSS-u - one će se koristiti zajedno sa Vue prilagođenom komponentom za kreiranje lepe aplikacije i obezbeđivanje najboljeg doživljaja korisnicima aplikacije.

U Poglavlju 9, „Kreiranje lepih aplikacija pomoću UI radnih okvira“, prikazani su popularni UI radni okviri. Čitalac će kreirati obrazac za registraciju korisnika pomoću UI radnih okvira Buefy, Vuetify i Ant-Design i njihovih koncepata dizajniranja. Cilj je da iz recepata u ovom poglavlju čitalac nauči kako da kreira lepe aplikacije pomoću UI radnog okvira.

U Poglavlju 10, „Implementacija aplikacije na Cloud platformama“, prikazano je kako se Vue aplikacija implementira na prilagođene hostove nezavisnih proizvođača, kao što su Vercel, Netlify i Google Firebase. Pomoću tih recepata u ovom poglavlju čitalac će naučiti kako da automatski implementira svoju aplikaciju pomoću integrisanih hooks spremišta i funkcija automatskog implementiranja.

U Poglavlju 11, „Profesionalna liga - direktive, pluginovi, SSR i još mnogo štošta“, istražene su napredne teme o Vueu, uključujući obrasce, najbolje tehnike, kako kreirati pluginove i direktive i kako koristiti radne okvire visokog nivoa za kreiranje aplikacija, kao što su Quasar i Nuxt.js.

Izvueite maksimum iz ove knjige

Ceo kod se ažurira, uvek kada ima razloga za to, u GitHub spremištu na adresi <http://bit.ly/3pB8oGm>

Potrebno je da imate instalirane Node.js 12+, Vue CLI ažuriran na najnoviju verziju i neki dobar uređivač. Ostali zahtevi biće predstavljeni u svakom receptu. Svi softverski zahtevi dostupni su za Windows, macOS i Linux.

Da biste razvili mobilne aplikacije za iOS, potreban vam je MacOS uređaj, radi pristupa Xcodeu i iOS simulatoru. U sledećoj tabeli su rezimirani svi zahtevi:

SOFTVER/HARDVER KOJI JE RAZMATRAN U KNJIZI	OS ZAHTEVI
Vue CLI 4.X	Windows/Linux/macOS
TypeScript 3.9.X	Windows/ Linux/macOS

SOFTVER/HARDVER KOJI JE RAZMATRAN U KNJIZI	OS ZAHTEVI
Quasar-CLI 1.X	Windows/Linux/macOS
Nuxt-CLI 3.X.X	Windows/Linux/macOS
Visual Studio Code 1.4.X i IntelliJ WebStorm 2020.2	Windows/Linux/macOS
Netlify-CLI	Windows/Linux/macOS
Vercel-CLI	Windows/Linux/macOS
Firebase-CLI	Windows/Linux/macOS
Node.js 12+-	Windows/Linux/macOS
Python 3	Windows/Linux/macOS
Xcode 11.4 i iOS Simulator	macOS

Ako koristite digitalnu verziju ove knjige, preporučujemo da sami unesete kod ili da mu pristupite pomoću spremišta GitHub (link je dostupan u sledećem odeljku). Ako kodu pristupite pomoću spremišta GitHub, izbeći ćete potencijalne greške povezane sa kopiranjem i „lepljenjem“ koda.

Preuzimanje datoteka sa primerima koda

Datoteke sa primerima koda za ovu knjigu možete da preuzmete sa našeg sajta:

<https://bit.ly/383uR97>

Kada je datoteka preuzeta, raspakujte ili ekstrahujte direktorijum, koristeći najnoviju verziju:

- WinRAR/7-Zip za Windows
- Zipeg/iZip/UnRarX za Mac
- 7-Zip/PeaZip za Linux

Paket koda za knjigu takođe se nalazi na GitHubu na adresi:

<http://bit.ly/3pB8oGm>

U slučaju da postoji njegovo ažuriranje, kod će biti ažuriran na postojećem GitHub spremištu.

Imamo i druge pakete koda iz našeg bogatog kataloga knjiga i video-zapisa, koji su dostupni na adresi <https://github.com/PacktPublishing/>.

Upotrebljene konvencije

Postoji veliki broj konvencija teksta koje su upotrebljene u ovoj knjizi.

CodeInText ukazuje na reči koda u tekstu, nazive tabele baze podataka, nazive direktorijuma, nazive datoteka, ekstenzije datoteka, nazive putanje, skraćene URL-ove, korisnički unos i Twitter postove. Evo i primera: „Priključite preuzetu datoteku slike diska `WebStorm-10*.dmg` kao drugi disk u vašem sistemu“.

Blok koda je prikazan na sledeći način:

```
<template>
<header>
  <div id="blue-portal" />
</header>
</header>
```

Svaki unos komandne linije ili ispis će biti prikazan na sledeći način:

```
$ npm run serve
```

Zadebljana slova ukazuju na novi termin, važnu reč ili reči koje vidite na ekranu. Na primer, reči u menijima ili okvirima za dijalog prikazane su u tekstu na sledeći način: „Kliknite na dugme **Email** da biste bili preusmereni na obrazac **Email Sign Up**“.



Napomene ili važna obaveštenja prikazani su ovako.



Saveti i trikovi su prikazani ovako.

Odeljci

U ovoj knjizi ćete naći nekoliko naslova koji se često pojavljuju („Priprema“, „Kako to uraditi...“, „Kako to funkcioniše...“, „Ima još...“ i „Takođe pogledajte“).

Da biste dobili jasna uputstva kako da dovršite recept, koristite ove odeljke na sledeći način:

Priprema

U ovom odeljku je ukazano šta možete da očekujete u receptu i opisano je kako da podesite bilo koji softver ili bilo koje preliminarne postavke neophodne za recept.

Kako to uraditi...

Ovaj odeljak sadrži korake koji su neophodni za praćenje recepta.

Kako to funkcioniše...

Ovaj odeljak se obično sastoji od detaljnog objašnjenja onoga o čemu je bilo reči u prethodnom odeljku.

Ima još...

Ovaj odeljak se sastoji od dodatnih informacija o receptu, radi boljeg upoznavanja recepta.

Takođe pogledajte

Ovaj odeljak sadrži korisne linkove do drugih korisnih informacija za recept.

Kontaktirajte sa nama

Povratne informacije od naših čitalaca su uvek dobrodošle.

Osnovne povratne informacije - Pošaljite e-poruku na adresu customercare@packtpub.com i u naslovu poruke napišite naslov knjige.

Štamparske greške - Iako smo preduzeli sve mere da bismo obezbedili tačnost sadržaja, greške su moguće. Ako pronađete grešku u ovoj knjizi, bili bismo zahvalni ako biste nam to prijavili. Posetite stranicu <http://www.packt.com/submit-errata>, izaberite knjigu, kliknite na link Errata Submission Form i unesite detalje.

Piraterija - Ako pronađete ilegalnu kopiju naših knjiga na Internetu, u bilo kojoj formi, molimo vas da nas o tome obavestite i da nam pošaljete adresu lokacije ili naziv veb sajta. Molimo vas, kontaktirajte sa nama na adresi copyright@packt.com i pošaljite nam link ka sumnjivom materijalu.

Ako ste zainteresovani da postanete autor - Ako postoji tema za koju ste specijalizovani i zainteresovani ste da pišete ili da sarađujete na nekoj od knjiga, pogledajte vodič za autore na adresi authors.packtpub.com.

Recenzija

Kada pročitate i upotrebite ovu knjigu, zašto ne biste napisali vaše mišljenje na sajtu sa kojeg ste je poručili? Potencijalni čitaoci tada mogu da upotrebe vaše mišljenje da bi odlučili o kupovini, mi u „Packtu“ možemo da znamo šta mislite o našim proizvodima, a naši autori mogu da vide povratne informacije o svojoj knjizi.

Više informacija o „Packtu“ naći ćete na sajtu packt.com.



Postanite član Kompjuter biblioteke

Kupovinom jedne naše knjige stekli ste pravo da postanete član Kompjuter biblioteke. Kao član možete da kupujete knjige u pretplati sa 40% popusta i učestvujete u akcijama kada ostvarujete popuste na sva naša izdanja.

Potrebno je samo da se prijavite preko formulara na našem sajtu.

Link za prijavu: <http://bit.ly/2TxeK5a>

Skenirajte QR kod
registrujte knjigu
i osvojite nagradu



1

Razumevanje Vuea 3 i kreiranje komponentata

U **Vueu 3** obezbeđeno je mnogo novih funkcija i izmena za programere, koje su dizajnirane da pomognu u razvoju i da poboljšaju ukupnu stabilnost, brzinu i održivost radnog okvira. Koristeći druge radne okvire i biblioteke kao inspiraciju, glavni Vue tim uspeo je da postigne odličan nivo apstrakcije na API-u u kojem Vue sada može da koristi svako, bez obzira da li je programer ili pomoćni programer.

U ovom poglavlju ćete naučiti kako da nadgradite Vue projekat na novu verziju i saznaćete više detalja o nekim novim Vue funkcijama, kao što su višestruki root elementi, novi mehanizam nasleđivanja atributa, upotreba reaktivnosti izloženog API-a izvan Vuea u drugoj aplikaciji i kreiranje komponente pomoću novog Composition API-a.

U ovom poglavlju obradićemo sledeće teme:

- novine u Vueu 3
- nadgradnja Vue 2 aplikacije na Vue 3
- kreiranje komponentata pomoću više root elemenata
- kreiranje komponentata pomoću nasleđivanja atributa
- upotreba reaktivnosti i uočljivog API-a izvan opsega primene Vuea
- kreiranje komponente pomoću Composition API-a

Šta je novo u Vueu 3

Možda se pitate zašto je nova verzija radnog okvira podigla toliku „prašinu“ na Internetu? Zamislite da se automobil kojim se vozite autoputem okrene za 360 stepeni i da zatim nastavite vožnju punom brzinom u istom smeru. To je savršen način da se opiše kako će Vue preći sa verzije 2 na verziju 3.

U ovom prvom delu poglavlja upoznaćete poboljšanja u Vueu, šta je dodato u radni okvir, šta se promenilo i kako će ta poboljšanja uticati na način kodiranja Vue aplikacije.

Poboljšanja radnog okvira

U ovom novom izdanju postoje brojna unapređenja radnog okvira Vue, koja su fokusirana na nastojanje da radni okvir postane što bolji. Evo nekih poboljšanja koja mogu uticati na svakodnevni razvoj i na to kako će korisnici i programeri koristiti radni okvir Vue.

Ispod „haube“

Spoljna „školjka“ izgleda isto kao i stara, ali „mašina“ je umetničko delo. U novoj verziji nema preostalog koda iz Vuea 2. Osnovni tim je izradio radni okvir od „temelja“ pomoću TypeScripta i ponovo je napisao sve prilagođeno maksimalnim performansama radnog okvira.

TypeScript je izabran da kreira održiviju bazu koda za Vue osnovni tim i zajednicu otvorenog koda i da poboljša funkcije automatskog dovršavanja, kao što su **IntelliSense** ili **typeahead**, koje obezbeđuju IDE-ovi i uređivači koda, bez potrebe za posebnim pluginovima i proširenjima.

Mehanizam za renderovanje

Za Vue 3 je razvijen novi mehanizam za renderovanje, pomoću novog algoritma za shadow DOM. Ovo novo renderovanje je podrazumevano potpuno izloženo „jezgru“ radnog okvira, pa ne mora da bude izvršeno. To omogućava nove implementacije potpuno nove funkcije renderovanja koja može da se injektuje u radni okvir i da zameni originalni mehanizam za renderovanje.

U ovoj novoj verziji Vuea novi kompajler šablona napisan je „od nule“. On koristi novu tehniku za manipulaciju keš memorijom i za upravljanje renderovanim elementima, a novi dizajnirani metod primenjuje se za kreiranje VNodesa.

Za manipulaciju keš memorijom primenjuje se novi metod za kontrolu položaja elementa koji može da bude dinamički element sa izračunatim podacima ili odgovor na funkciju koja može biti mutirana.

Vue osnovni tim kreirao je explorer u kojem možete da vidite kako novi kompajler šablona generiše konačnu funkciju `render`. Explorer možete da pogledate na adresi <https://vue-next.template-explorer.netlify.app/>.

Izloženi API-i

Zahvaljujući svim ovim modifikacijama, mogu se renderovati svi izloženi Vue API-i tako da se koriste u datotekama izvan opsega primene Vuea. Možete da koristite Vue reaktivnost ili shadow DOM u aplikaciji React, bez potrebe da se Vue aplikacija prikazuje unutar aplikacije React. Ova upotrebljivost omogućava pretvaranje Vuea u svestraniji radni okvir koji se može koristiti bilo gde, a ne samo u frontend razvoju.

Nove prilagođene komponente

Vue 3 uvodi tri nove prilagođene komponente koje programer može koristiti za rešavanje starih problema. Ove komponente su bile prisutne u Vueu 2, ali kao nezavisni pluginovi i proširenja. Sada ih je kreirao Vue osnovni tim i dodao ih u Vue osnovni radni okvir.

Fragments API

U Vueu 2 uvek je bio potreban nadređeni čvor koji omotava komponente unutar komponentata jedne datoteke. Ovo je prouzrokovano načinom na koji je konstruisan mehanizam za obradu Vuea 2, koji zahteva root element na svakom čvoru.

U Vueu 2 morali smo da koristimo element omotača, koji obuhvata elemente koji će biti renderovani. U primeru imamo HTML element `div` koji omotava dva HTML podređena elementa `p`, tako da možemo da koristimo više elemenata na stranici:

```
<template>
  <div>
    <p>This is two</p>
    <p>children elements</p>
  </div>
</template>
```

Sada u Vueu 3 možemo da deklariramo bilo koji broj root elemenata u komponentama jedne datoteke, bez potrebe za specijalnim pluginovima, pomoću novog Fragments API-a koji će upravljati višestrukim root elementima. Ovaj API pomaže u održavanju čistijeg konačnog koda za korisnika, bez potrebe za praznim „ljuska-ma“ samo za omotavanje elemenata:

```
<template>
  <p>This is two</p>
  <p>root elements</p>
</template>
```

Kao što ste videli u Vue 3 kodu, mogli smo da upotrebimo dva osnovna HTML elementa `p`, bez potrebe za elementom omotača.

Teleport

Komponenta Teleport, takođe poznata kao komponenta Portal, kao što sam naziv ukazuje, omogućava da element pređe iz jedne komponente u drugu. To u početku može izgledati čudno, ali komponenta Teleport se može primeniti na okvir za dijalog, prilagođene menije, upozorenja, „značke“ i mnoge druge prilagođene UI-e koji treba da budu prikazani na posebnim mestima.

Zamislite komponentu zaglavlja u kojoj želite prilagođeni slot da biste mogli da postavite komponente:

```
<template>
  <header>
    <div id="blue-portal" />
  </header>
</header>
```

Ako želite da u ovom zaglavlju prikazete prilagođeno dugme, ali da ga pozovete sa stranice, treba samo da izvršite sledeći kod:

```
<template>
  <page>
    <Teleport to="blue-portal">
      <button class="orange-portal">Cake</button>
    </Teleport>
  </page>
</template>
```

Sada će dugme biti prikazano u zaglavlju, ali kod će biti izvršen na stranici, čime će se omogućiti pristup opsegu stranice.

Suspense

Kada čekanje podataka traje duže nego što biste želeli, kako bi bilo da pokažete prilagođeni loader za korisnika? To je sada moguće bez prilagođenog koda; Vue će to uraditi automatski. Komponenta `Suspense` upravljaće ovim procesom pomoću podrazumevanog prikaza nakon što su podaci učitani i pomoću rezervnog prikaza dok se podaci učitavaju.

Možete napisati poseban omotač poput ovog:

```
<template>
  <Suspense>
    <template #default>
      <data-table />
    </template>
    <template #fallback>
      <loading-gears />
    </template>
  </Suspense>
</template>
```

Novi Vue Composition API će „razumeti“ trenutno stanje vaše komponente, pa će moći da razlikuje da li se komponenta učitava ili da li je spremna za prikaz.

Izmene API-a

U Vueu 3 su izvršene neke izmene API-a koje su bile neophodne da bi Vue API bio očišćen i da bi bila pojednostavljena implementacija. Neke od izmena su korenite promene, a druge se odnose na plaginove. Međutim, ne brinite - implementacija Vue 2 objekta nije uklonjena i koristiće se i dalje. Ovaj metod deklarisanja je jedan od razloga zbog kojeg mnogi programeri biraju Vue, a ne druge radne okvire.

U Vueu 3 će se dogoditi neke korenite promene, o kojima treba da saznate što više detalja. Razmotrićemo najvažnije korenite promene koje će biti uvedene u Vueu 3 i o tome kako da se suočite sa njima.

U Vueu 3 se uvodi novi metod kreiranja komponentata - Composition API. Ovaj metod će poboljšati održivost vašeg koda i obezbediće pouzdaniji kod u kojem ćete imati na raspolaganju punu moć TypeScripta.

Neke manje korenite promene

Postoje neke manje korenite promene u Vueu 3, a koje treba pomenuti.

Ove promene se odnose na jedan metod koji smo ranije koristili za pisanje koda, a koji je zamenjen u Vueu 3. Za te promene nije potreban herkulovski napor, ali morate da saznate više detalja o njima.

Zbogom filteri, zdravo filteri! Vue filter API

Način na koji smo koristili `filters` u Vueu 2 više nije dostupan. Vue filter je uklonjen iz API-a. Ova promena je izvršena da bi bio pojednostavljen i ubrzan proces renderovanja. Svi filteri su funkcije koje primaju i vraćaju znakovni niz.

U Vueu 2 smo koristili filtere poput sledećeg:

```
{{ textString | filter }}
```

U Vueu 3 samo treba da prosledimo funkciju za manipulisanje znakovnim nizom:

```
{{ filter(textString) }}
```

Autobus je upravo krenuo sa stanice - Event bus API

U Vueu 2 smo mogli da iskoristimo moć globalnog Vue objekta za kreiranje nove Vue instance i da tu instancu koristimo kao event bus koji može da prenosi poruke između komponentata i funkcija, bez ikakvih muka. Treba samo da objavimo event bus i da se pretplatimo na njega i sve će biti savršeno.

Ovo je bio dobar način za prenos podataka između komponentata, ali je bio „antio-brazac“ pristupa za Vue radni okvir i komponente. Pravilan način prenosa podataka između komponenta u Vueu je korišćenje komunikacije „nadređeni-podređeni“ ili korišćenje upravljanja stanjem, koje je poznato i kao arhitektura koja je vođena stanjem.

Iz Vuea 3 su uklonjeni metodi `$on`, `$off` i `$once`. Da biste sada koristili strategiju event bus, preporučuje se upotreba nezavisnog plagina ili radnog okvira, kao što je mitt (<https://github.com/developit/mitt>).

Nema više globalnog Vuea - mounting API

U Vueu 2 smo navikli da uvozimo Vue, a pre montiranja aplikacije da pomoću globalne instance Vue dodamo `plugins`, `filters`, `components`, `router` i `store`. To je bila dobra tehnika pomoću koje smo mogli da dodamo bilo šta u Vue instancu, a da, pri tom, nismo morali da priključimo nešto direktno u montiranu aplikaciju. To je funkcionisalo ovako:

```
import Vue from 'vue';
import Vuex from 'vuex';
import App from './App.vue';

Vue.use(Vuex);
const store = new Vuex.store({});

new Vue({
  store,
  render: (h) => h(App),
}).$mount('#app');
```

U Vueu 3 to više nije moguće. Moramo direktno da priključimo svaki `component`, `plugin`, `store` i `router` direktno u montiranu instancu:

```
import { createApp } from 'vue';
import { createStore } from 'vuex';
import App from './App.vue';

const store = createStore({});

createApp(App)
  .use(store)
  .mount('#app');
```

Pomoću ovog metoda možemo da kreiramo različite Vue aplikacije u istoj globalnoj aplikaciji, bez pluginova, skladišta ili rutera aplikacija koje se međusobno ne mešaju.

v-model, v-model, v-model - višestruki v-model

Kada smo razvijali komponentu jedne datoteke, koristili smo direktivu `v-model` i opciju `.sync` za drugu izmenu ažuriranja. Znači da koristimo mnogo prilagođenih emitora događaja i ogromni payload za obradu podataka unutar komponente.



U ovoj korenitoj promeni je uvedena kolateralna korenita promena rezultirala uklanjanjem svojstva `model` (<https://vuejs.org/v2/api/#model>) iz Vue API-a. Ovo svojstvo se koristi u prilagođenim komponentama koje su radile isto što sada radi nova direktiva `v-model`.

Novi način upotrebe direktive `v-model` promeniće način na koji funkcioniše sintaksni „šećer“. Da bismo u Vueu 2 koristili direktivu `v-model`, morali smo da kreiramo komponentu koja očekuje da primi `props` kao `"value"`, a kada je došlo do promene, morali smo da emitujemo događaj `'input'`, kao u sledećem kodu:

```
<template>
  <input
    :value="value"
    @input="$emit('input', $event)"
  />
</template>
<script>
export default {
  props: {
    value: String,
  },
}
</script>
```

U Vueu 3, da bi sintaksni „šećer“ funkcionisao, komponenta će primiti svojstvo `props` i emiter događaja će se promeniti. Sada komponenta očekuje `props` pod nazivom `modelValue` i emituje događaj `'update:modelValue'`, kao u sledećem kodu:

```
<template>
  <input
    :modelValue="modelValue"
    v-on:['update:modelValue']="<code>$emit('update:modelValue', $event)"
  />
</template>
<script>
export default {
  props: {
    modelValue: String,
  },
}
</script>
```

Razumevanje korenite promene direktive `v-model` je prvi korak u razumevanju kako će funkcionisati novi metod višestruke direktive `v-model`.

Da bismo kreirali više komponentata `v-model`, treba da kreiramo različita svojstva `props` pomoću naziva direktive modela po želji i da emitujemo događaje `'update:value'` u kojima je vrednost naziv direktive modela:

```
<script>
export default {
  props: {
    name: String,
    email: String,
  },
  methods: {
    updateUser(name, email) {
      this.$emit('update:name', name);
      this.$emit('update:email', email);
    }
  }
}
</script>
```

U komponenti u kojoj želimo da koristimo više direktiva `v-model` koristićemo sledeći kod:

```
<template>
  <custom-component
    v-model:name="name"
    v-model:email="email"
  />
</template>
```

Komponenta će imati svaku direktivu `v-model` koja je ograničena na događaj koji emituje podređeni element. U ovom slučaju podređena komponenta emituje `'update:email'` (nadređenu komponentu) da bi mogla da koristi direktivu `v-model` sa modifikatorom `e-pošte`. Na primer, `v-model:email` možete koristiti za kreiranje dvosmernog vezivanja komponente i podataka.

Composition API

Ovo je jedna od najočekivanijih Vue 3 funkcija. Composition API je novi metod za kreiranje Vue komponentata, sa optimizovanim načinom pisanja koda i obezbeđivanjem pune podrške za proveru TypeScript tipa u našoj komponenti. Ovaj metod organizuje kod na jednostavniji i efikasniji način.

U ovom novom metodu deklarisanja Vue komponente postoji samo svojstvo `setup` koje će se izvršiti i vratiće sve što je našoj komponenti potrebno da bi bila izvršena, kao u sledećem primeru:

```
<template>
  <p @click="increaseCounter">{{ state.count }}</p>
</template>
<script>
import { reactive, ref } from 'vue';

export default {
  setup() {
    const state = reactive({
      count: ref(0)
    });

    const increaseCounter = () => {
      state.count += 1;
    }

    return { state, increaseCounter }
  }
}
</script>
```

Treba da uvezete `reactivity` API iz Vue jezgra da biste ga omogućili u svojstvu podataka tipa objekta - u ovom slučaju `state`. `Ref` API omogućava reaktivnost u osnovnoj vrednosti tipa, kao što je `count`, koji je broj.

Na kraju, funkcije se mogu deklarirati unutar funkcija `setup` i proslediti vraćenom objektu. Onda je sve dostupno u delu `<template>`.

Sada ćemo preći na neke recepte.

Tehnički zahtevi

U ovom poglavlju ćemo koristiti **Node.js** i **Vue-CLI**.



Ako ste korisnik Windowsa, morate da instalirate NPM paket `windows-build-tools` da biste mogli da instalirate neophodne pakete koji su prikazani u nastavku. Da biste to uradili, otvorite PowerShell kao administrator i izvršite sledeću komandu:

```
>npminstall -g windows-build-tools.
```


Da biste instalirali Vue-CLI, potrebno je da otvorite Terminal (macOS ili Linux) ili CommandPrompt/PowerShell (Windows) i da izvršite sledeću komandu:

```
> npm install -g @vue/cli @vue/cli-service-global
```

Kreiranje osnovne datoteke

U svim receptima u ovom poglavlju koristićemo osnovni obrazac koji ćemo sada da kreiramo. Obavezno sledite ove korake za kreiranje datoteke pre nego što pokrenete primer iz recepta:

1. Kreirajte novu `.html` datoteku u bilo kojoj fascikli i otvorite je.
2. Kreirajte `html` tag i dodajte HTML element `head` kao podređeni element. U HTML elementu `head` dodajte HTML element `script` pomoću atributa `src`, koji je definisan kao `http://unpkg.com/vue@next`:

```
<html>
  <head>
    <script src="https://unpkg.com/vue@next"></script>
  </head>
</html>
```

3. Kreirajte HTML element `body` kao „potomak“ HTML elementa `head`. Unutar HTML elementa `body` dodajte HTML element `div` pomoću atributa `id` koji je definisan kao `"app"`:

```
<body>
  <div id="app">
  </div>
</body>
```

4. Na kraju kreirajte HTML element `script`, sa praznim sadržajem, kao „potomak“ ili element HTML elementa `div`. Ovde ćemo postaviti kod za recepte:

```
<script></script>
```

Nadgradnja Vue 2 aplikacije na Vue 3

Nadgradnja vašeg projekta sa Vue 2 na Vue 3 ponekad se može izvršiti automatski, ali u drugim slučajevima to treba učiniti ručno. A kako ćete ga nadgraditi zavisi od toga koliko ste duboko „zaronili“ u Vue API u vašoj aplikaciji.

U projektima koje je Vue-CLI izradio i kojima je upravljao, ovaj proces će biti izrađen neprimetno i imaće jednostavniji pristup, u poređenju sa projektima koji koriste CLI prilagođeni omotač radnog okvira.

U ovom receptu ćete naučiti kako da nadgradite svoju aplikaciju pomoću Vue-CLI-a i kako da ručno nadgradite projekat i zavisnosti.

Priprema

Preduslov za ovaj recept je sledeći:

- Node.js 12+

Globalni objekti Node.js koji su potrebni su sledeći:

- @vue/cli
- @vue/cli-service-global

Kako to uraditi...

Da biste nadgradili svoj Vue 2 projekat na Vue 3, moraćete da podelite nadgradnju na različite delove. Treba da nadgradite sam radni okvir, a zatim komponente ekosistema, kao što su vue-router i vuex, i paker koji na kraju sve spaja.



Nadgradnja radnog okvira se isporučuje sa korenitim promenama. U ovoj knjizi su predstavljene neke korenite promene u odeljku „Šta je novo u Vueu 3“ iz ovog poglavlja i druge promene koje se mogu pojaviti u naprednijoj šemi API-a. Morate ručno da ažurirate radni okvir i da proverite da li su vaše komponente validne za nadgradnju u radnom okviru.

Korišćenje Vue-CLI-a za nadgradnju projekta

Kada koristite najnoviju verziju Vue-CLI-a, moći ćete da koristite Vue 3 u svom projektu odmah iz radnog okvira i da nadgradite svoj trenutni projekat na Vue 3.

Da biste nadgradili Vue-CLI na najnoviju verziju, potrebno je da otvorite Terminal (macOS ili Linux) ili CommandPrompt/PowerShell (Windows) i da izvršite sledeću komandu:

```
> npm install @vue/cli-service@latest
```

Ručna nadgradnja projekta

Da biste ručno nadgradili projekat, prvo ćete morati da nadgradite zavisnosti projekta na najnovije verzije. U Vueu 3 ne možete da koristite staru verziju plugina za Vue ekosistem. Da biste ručno nadgradili projekat, izvršite sledeće korake:

1. Morate da nadgradite Vue radni okvir, plugin ESLint (od kojeg Vue zavisi) i `vue-loader` za paker. Da biste izvršili nadgradnju, morate da otvorite Terminal (macOS ili Linux) ili CommandPrompt/PowerShell (Windows) i da izvršite sledeću komandu:

```
> npm install vue@next eslint-plugin-vue@next vue-loader@next
```

2. Morate da dodate novu Vue komponentu kompajlera jedne datoteke kao zavisnost projekta. Da biste je instalirali, potrebno je da otvorite Terminal (macOS ili Linux) ili CommandPrompt/PowerShell (Windows) i da izvršite sledeću komandu:

```
> npm install @vue/compiler-sfc@latest
```

3. Ako u svom projektu koristite jedinične testove i paker `@vue/test-utils`, moraćete da nadgradite i ovu zavisnost. Da biste je nadgradili, morate da otvorite Terminal (macOS ili Linux) ili CommandPrompt/PowerShell (Windows) i da izvršite sledeću komandu:

```
> npm install @vue/test-utils@next @vue/server-test-utils@latest
```

4. Ako koristite komponentu `vue-router` za pluginove Vue ekosistema, moraćete i nju da nadgradite. Da biste izvršili nadgradnju, morate da otvorite Terminal (macOS ili Linux) ili CommandPrompt/PowerShell (Windows) i da izvršite sledeću komandu:

```
> npm install vue-router@next
```

5. Ako vaša aplikacija koristi komponentu `vuex` kao podrazumevano upravljanje stanjem, moraćete i nju da nadgradite. Da biste izvršili nadgradnju, morate da otvorite Terminal (macOS ili Linux) ili CommandPrompt/PowerShell (Windows) i da izvršite sledeću komandu:

```
> npm install vuex@next
```

Izmena početnih datoteka

Zbog nove verzije paketa, morate da izmenite početne datoteke. U Vue projektu koji je kreiran pomoću početnog kompleta Vue-CLI pronaći ćete datoteku koja se zove `main.js` ili `main.ts`. Ako koristite TypeScript, ova datoteka se nalazi u fascikli `src`. Sada sledite ova uputstva:

1. Otvorite datoteku `main.js` u fascikli `src` vašeg projekta. Na vrhu datoteke u kojoj se uvoze paketi videćete sledeći kod:

```
import Vue from 'vue';
```

Ovaj kod morate da izmenite u novi Vue izloženi API metod. Da bismo to uradili, treba da uvezete `createApp` iz Vue paketa na sledeći način:

```
import { createApp } from 'vue';
```

2. Uklonite globalnu definiciju statičkog atributa `Vue` `Vue.config.productionTip` iz koda.
3. Funkciju montiranja vaše aplikacije treba promeniti. Stari API izgleda ovako:

```
new Vue({
  router,
  store,
  render: (h) => h(App),
}).$mount('#app');
```

Stari API treba promeniti u novi API `createApp` na sledeći način:

```
createApp(App)
  .use(router)
  .use(store)
  .mount('#app')
```

4. Otvorite datoteku instanciranja skladišta `vuex` (obično se ova datoteka nalazi u fascikli `src/store` i naziva se `store.js` ili `index.js`).
5. Promenite kreiranje skladišta iz instanciranja nove `vuex` klase u novi API `createStore`. Instanca klase `vuex v3` može izgledati ovako:

```
import Vue from 'vue';
import Vuex from 'vuex';

Vue.use(Vuex);

export default new Vuex.Store({
```

```
state: { /* ... */ },
mutations: { /* ... */ },
actions: { /* ... */ },
getters: { /* ... */ },
modules: { /* ... */ },
});
```

Morate da zamenite njegov sadržaj API-em `createStore` koji bi mogao izgledati ovako:

```
import { createStore } from 'vuex';

export default createStore({
  state: { /* ... */ },
  mutations: { /* ... */ },
  actions: { /* ... */ },
  getters: { /* ... */ },
  modules: { /* ... */ },
});
```

6. U ekosistemu `vue-router` moraćete da zamenite stari API iz kreiranja rutera novim. Da biste to uradili, otvorite datoteku za kreiranje rutera (u fascikli `src/router` koja se, obično, naziva `router.js` ili `index.js`).
7. Na kraju, u datoteci kreiranja zamenite staru instancu klase `vue-router` novim API-em `createRouter`. Klasa v3 instance `vue-router` može izgledati ovako:

```
import Vue from 'vue';

import VueRouter from 'vue-router';

Vue.use(VueRouter);
export default new VueRouter({
  routes: [{
    path: '/',
    name: 'HomePage',
    component: () => import('pages/home'),
  }]
});
```

Takođe ćete morati da zamenite instanciranje `newVueRouter` novim API-ima `createRouter` i `createWebHistory`, kao u sledećem primeru:

```
import {
  createRouter,
  createWebHistory,
} from 'vue-router';
```

```
Vue.use(VueRouter);

export default createRouter({
  history: createWebHistory(),
  routes: [{
    path: '/',
    name: 'HomePage',
    component: () => import('pages/home'),
  }]
});
```

Kako to funkcioniše...

U procesu nadgradnje Vue je obezbedio dva načina da se ažurira projekat. Prvi način je upotreba plagina Vue-CLI, koji pokušava da automatizuje skoro sve procese i izmene potrebne za nadgradnju.

Drugi način je ručna nadgradnja projekta. Ovaj metod zahteva od programera da nadgradi sve zavisnosti na najnoviju verziju, da instalira `@vue/compiler-sfc`, novi kompajler komponenata jedne datoteke, i da zameni datoteke unosa za Vue aplikaciju, ruter i skladište novim API-em.

Nakon izmena u početnoj strukturi projekta, programer treba da proveri komponente da bi utvrdio da li neke promene krše Vue 3, da refaktoriše komponentu na nove Vue 3 API-e i da ukloni zastarele API-e iz Vuea 2.

Kreiranje komponenata pomoću više root elemenata

U Vueu 3 je moguće kreirati komponente pomoću više root elemenata, bez elemenata omotavanja (ova opcija je poznata i kao fragment).

U Reactu je to već dugo moguće, ali u Vueu morate da koristite prilagođene nezavisne plaginove, kao što je `vue-fragment` (<https://github.com/Thunberg087/vue-fragment>), da biste koristili ovu funkciju.

U ovom receptu ćete naučiti kako da kreirate komponentu pomoću više root elemenata i kako se ona može koristiti u delu `<template>` i u funkciji `render`.

Kako to uraditi...

U ovom receptu ćete kreirati dva primera komponente višestrukog root elementa - jedan pomoću strukture `<template>`, a drugi pomoću funkcije `render`. Da biste to uradili, ovaj recept će biti podeljen na dva dela.

Kreiranje komponente pomoću strukture `<template>`

Da bismo koristili strukturu `<template>` u primeru, koristićemo svojstvo `template` objekta `Vue` u kojem možemo kao vrednost da prosledimo znakovni niz ili znakovni niz šablona, koji će `Vue` skript interpolirati i prikazati na ekranu:

1. Pomoću osnovnog primera iz odeljka „Kreiranje osnovne datoteke“ kreirajte novu datoteku `template.html`, a zatim je otvorite.
2. U praznom HTML elementu `<script>` kreirajte konstante `defineComponent` i `createApp` objektnim destrukuiranjem globalne konstante `Vue`:

```
const {
  defineComponent,
  createApp,
} = Vue;
```

3. Kreirajte konstantu `component` definisanu kao metod `defineComponent`, tako što ćete proslediti JavaScript objekat kao argument pomoću svojstava `data`, `methods` i `template`.

```
const component = defineComponent({
  data: () => ({}),
  methods: {},
  template: ``
});
```

4. U svojstvu `data` definišite konstantu kao singularnu funkciju, tako što ćete vratiti JavaScript objekat kao rezultat pomoću svojstva `count` i podrazumevane vrednosti `0`:

```
data: () => ({
  count: 0
}),
```

- U svojstvu `methods` kreirajte svojstvo `addOne`, a to je funkcija koja će povećati vrednost `count` za 1:

```
methods: {
  addOne() {
    this.count += 1;
  },
},
```

- U svojstvu `template` u znakovnom nizu šablona kreirajte HTML element `h1` sa naslovom. Zatim, kreirajte, kao „potomak“, HTML element `button` pomoću oslušivača događaja koji je „vezan“ za događaj `click`, tako što ćete pokrenuti funkciju `addOne` kada se izvrši:

```
template: `
  <h1>
    This is a Vue 3 Root Element!
  </h1>
  <button @click="addOne">
    Pressed {{ count }} times.
  </button>
`
```

- Na kraju, pozovite funkciju `createApp`, tako što ćete proslediti konstantu `component` kao argument. Zatim, izradite lanac prototipova funkcije `mount` i, kao argument funkcije, dodajte atribut `div` HTML elementa (`"#app"`):

```
createApp(component)
  .mount('#app');
```

Kreiranje komponente pomoću funkcije `render`

Da bismo koristili strukturu `<template>` u primeru, upotrebićemo svojstvo `template` objekta `Vue` u kojem možemo kao vrednost da prosledimo znakovni niz ili znakovni niz šablona, koji će `Vue` skript interpolirati i prikazati na ekranu:

- Pomoću osnovnog primera iz odeljka „Kreiranje osnovne datoteke“ kreirajte novu datoteku `render.html`, a zatim je otvorite.

- U praznom HTML elementu `<script>` kreirajte konstante funkcija koje će se koristiti pomoću metoda destrukuiranja objekata, tako što ćete pozvati metode iz globalne Vue konstante `defineComponent`, `h` i `createApp`:

```
const {
  defineComponent,
  h,
  createApp,
} = Vue;
```

- Kreirajte konstantu `component` definisanu kao metod `defineComponent`, tako što ćete proslediti JavaScript objekat kao argument sa svojstvima `data`, `methods` i `render`:

```
const component = defineComponent({
  data: () => ({}),
  methods: {},
  render() {},
});
```

- U svojstvu `data` definišite konstantu kao singularnu funkciju, tako što ćete vratiti JavaScript objekat kao rezultat pomoću svojstva `count` i podrazumevane vrednosti 0:

```
data: () => ({
  count: 0
}),
```

- U svojstvu `methods` kreirajte svojstvo `addOne`, a to je funkcija koja će povećati vrednost `count` za 1:

```
methods: {
  addOne() {
    this.count += 1;
  },
},
```

- U svojstvu `render` izvršite sledeće korake:
 - Kreirajte konstantu koja se zove `h1` i definišite je kao funkciju `h`, tako što ćete proslediti `h1` kao prvi argument i naslov koji će biti korišćen kao drugi argument.
 - Kreirajte konstantu `button` koja će biti `h` funkcija, tako što ćete proslediti „`button`“ kao prvi argument, JavaScript objekat pomoću svojstva `onClick` sa vrednošću `this.addOne` kao drugim argumentom i sadržajem `button` kao trećem argumentom.

- o Vratite niz kao rezultat pomoću prve vrednosti kao `h1` konstante i pomoću druge vrednosti kao konstante `button`:

```
render() {
  const h1 = h('h1', 'This is a Vue 3 Root Element!');
  const button = h('button', {
    onClick: this.addOne,
  }, `Pressed ${this.count} times.`);

  return [
    h1,
    button,
  ];
},
```

7. Na kraju, pozovite funkciju `createApp`, tako što ćete da prosledite konstantu `component` kao argument, da izradite lanac prototipova koji povezuje funkciju `mount` i da prosledite atribut `id("#app")` HTML elementa `div` kao argument funkcije:

```
createApp(component)
  .mount('#app');
```

Kako to funkcioniše...

API za kreiranje nove komponente Vue treba da izvrši funkcija `defineComponent`, a JavaScript objekat koji se prosleđuje kao argument ima skoro istu strukturu kao i stara struktura u Vueu 2. U primerima smo koristili ista svojstva `data`, `render` i `methods` koja su prisutna u Vueu 2.

U primeru strukture `<template>` nismo morali da kreiramo element omotača da bismo enkapsulirali sadržaj komponente aplikacije i mogli smo da direktno imamo dva root elementa u komponenti.

U primeru funkcije `render` dešava se isto, ali u poslednjem primeru je korišćen novi izloženi API `h` koji više nije parametar funkcije `render`. Korenita promena je bila prisutna u primeru; prilikom kreiranja dugmeta morali smo da koristimo svojstvo `onClick` unutar objekta JavaScript podataka, a ne svojstvo `on` zajedno sa metodom `click`, zbog nove strukture podataka `VNodea` Vuea 3.

Kreiranje komponenata pomoću nasleđivanja atributa

Od Vuea 2 bilo je moguće koristiti atribut nasleđivanja u komponentama, ali u Vueu 3 je nasleđivanje atributa poboljšano pomoću pouzdanijeg API-a za upotrebu u komponentama.

Nasleđivanje atributa u komponentama je obrazac koji omogućava brži razvoj prilagođenih komponenata zasnovanih na HTML elementima (poput prilagođenih unosa, dugmadi, omotača teksta ili linkova).

U ovom receptu ćemo kreirati prilagođenu ulaznu komponentu pomoću nasleđivanja atributa koje ćemo primeniti direktno na ulazni HTML element.

Kako to uraditi....

Ovde ćemo kreirati komponentu koja će imati potpuno nasleđivanje atributa u izabranom elementu na DOM stablu:

1. Pomoću osnovnog primera iz odeljka „Kreiranje osnovne datoteke“ kreirajte novu datoteku `component.html`, a zatim je otvorite.
2. U praznom HTML elementu `<script>` kreirajte konstante `defineComponent` i `createApp` objektnim destrukuiranjem globalne konstante `Vue`:

```
const {
  defineComponent,
  createApp,
} = Vue;
```

3. Kreirajte konstantu `nameInput` definisanu kao metod `defineComponent`, tako što ćete proslediti JavaScript objekat kao argument sa svojstvima `name`, `props`, `template` i `inheritAttrs`. Zatim, definišite vrednost `inheritAttrs` kao `false`:

```
const nameInput = defineComponent({
  name: 'NameInput',
  props: {},
  inheritAttrs: false,
  template: ``
});
```

4. U svojstvu `props` dodajte svojstvo `modelValue` i definišite ga kao `String`:

```
props: {
  modelValue: String,
},
```

5. U svojstvu `template` unutar znakovnog niza `template` morate uraditi sledeće:

- Kreirajte HTML element `label` i dodajte HTML element `input` kao podređeni element.
- U HTML elementu `input` definišite direktivu `v-bind` kao JavaScript objekat pomoću destrukuirane vrednosti `this.$attrs`.
- Definišite promenljivu atributa `value` kao `modelValue` primljenog svojstva `modelValue`.
- Postavite `type` atributa `input` kao „text“.
- U osluškivaču događaja `change` dodajte anonimnu funkciju koja prima event kao argument, a zatim emituje (`emit`) događaj koji se zove „update:modeValue“ sa korisnim podacima `event.target.value`:

```
template: `  
  <label>  
    <input  
      v-bind="{  
        ...$attrs,  
      }"  
      :value="modelValue"  
      type="text"  
      @change="(event) => $emit('update:modelValue',  
                                event.target.value)"  
    />  
  </label>`
```

6. Kreirajte konstantu `appComponent` definisanu kao metod `defineComponent`, tako što ćete proslediti JavaScript objekat kao argument sa svojstvima `data` i `template`:

```
const component = defineComponent({  
  data: () => ({}),  
  template: ``,  
});
```

- U svojstvu `data` definišite konstantu kao singularnu funkciju, tako što ćete vratiti JavaScript objekat kao rezultat pomoću svojstva `name` i podrazumevane vrednosti `0`:

```
data: () => ({
  name: ''
}),
```

- U svojstvu `template` unutar znakovnog niza `template` morate uraditi sledeće:

- Kreirajte komponentu `NameInput` pomoću direktive `v-model` koja je povezana sa svojstvom podataka `name`.
- Kreirajte atribut `style` pomoću vrednosti „`border:0; border-bottom:2px solid red;`“.
- Kreirajte atribut `data-test` pomoću vrednosti „`name-input`“:

```
template: `
<name-input
  v-model="name"
  style="border:0; border-bottom: 2px solid red;"
  data-test="name-input"
/>`
```

- Kreirajte konstantu `app` i definišite je kao funkciju `createApp`, tako što ćete proslediti konstantu `component` kao argument. Zatim, pozovite funkciju `app.component`, tako što ćete proslediti kao prvi argument naziv komponente koju želite da registrujete, a kao drugi argument komponentu. Na kraju, pozovite funkciju `app.mount`, tako što ćete proslediti `"#app"` kao argument:

```
const app = createApp(component);
app.component('NameInput', nameInput);
app.mount('#app');
```

Kako to funkcioniše...

Da bismo u Vueu 3 kreirali komponentu, treba da izvršimo funkciju `defineComponent`, tako što ćemo proslediti JavaScript kao argument. Ovaj objekat ima skoro istu strukturu kao što je stara struktura u Vueu 2. U primerima smo koristili ista svojstva `data`, `render`, `methods` i `template` koja su prisutna u Vueu 2.

Koristili smo svojstvo `inheritAttrs` da bismo blokirali automatsku primenu atributa na sve elemente u komponentama, tako što smo ih primenili samo na element sa direktivom `v-bind` i sa destrukuiranim objektom `this.$attrs`.

Da bismo registrovali komponentu u aplikaciji Vue, prvo smo kreirali aplikaciju pomoću API-a `createApp`, a zatim smo izvršili funkciju `app.component` za globalnu registraciju komponente u aplikaciji, pre nego što smo je renderovali.

Upotreba reaktivnosti i uočljivog API-a izvan opsega Vuea

U Vueu 3 sa izloženim API-ima možemo koristiti Vue reaktivnost i reaktivne promenljive, bez potrebe za kreiranjem Vue aplikacije. Ovo omogućava backend i frontend programerima da u potpunosti iskoriste Vue API `reactivity` u svojoj aplikaciji.

U ovom receptu ćemo kreirati jednostavnu JavaScript animaciju pomoću API-a `reactivity` i `watch`.

Kako to uraditi...

Ovde ćete kreirati aplikaciju koja će koristiti Vue API `reactivity` za renderovanje animacije na ekranu:

1. Pomoću osnovnog primera iz odeljka „Kreiranje osnovne datoteke“ kreirajte novu datoteku `reactivity.html`, a zatim je otvorite.
2. U tag `<head>` dodajte novi tag `<meta>`, zajedno sa skupom atributa `charset` koji je definisan kao `"utf-8"`:

```
<meta charset="utf-8"/>
```

3. Iz taga `<body>` uklonite HTML element `div#app` i kreirajte HTML element `div` pomoću ID-ja koji je definisan kao `marathon` i atributa `style` koji je definisan kao `"font-size: 50px; "`:

```
<div
  id="marathon"
  style="font-size: 50px;"
>
</div>
```

- U praznom HTML elementu `<script>` kreirajte konstante funkcija koje će se koristiti pomoću metoda destrukuiranja objekta, pri čemu pozivate `reactivity` i metode `watch` iz globalne konstante `Vue`:

```
const {
  reactive,
  watch,
} = Vue;
```

- Kreirajte konstantu koja se zove `mod` definisanu kao funkciju, koja prima dva argumenta `a` i `b`, pa će zatim biti vraćena aritmetička operacija `a` modulo `b`:

```
const mod = (a, b) => (a % b);
```


- Kreirajte konstantu `maxRoadLength` pomoću vrednosti 50. Zatim, kreirajte konstantu imenovanu `competitor` pomoću vrednosti kao funkcije `reactivity`, tako što ćete proslediti JavaScript objekat kao argument pomoću svojstva `position` definisanog kao 0 i svojstva `speed` definisanog kao 1:

```
const maxRoadLength
= 50;
const competitor =
  reactive({
    position: 0,
    speed: 1,
  });
```

- Kreirajte funkciju `watch`, tako što ćete proslediti anonimnu funkciju kao argument. Unutar funkcije uradite sledeće:

- Kreirajte konstantu `street` i definišite je kao `Array` veličine `maxRoadLength` i popunite je znakovima `_,_`.
- Kreirajte konstantu `marathonEl` i definišite je kao HTML DOM čvor `#marathon`.
- Izaberite element `street`

u indeksu niza `competitor.position` i definišite ga kao „ „

ako je broj za `competitor.position` paran ili kao “ „, ako je taj broj neparan.

- Definišite `marathonEl.innerHTML` kao „“ i kao `street.reverse().Join(, ‘)`:

Emotikoni koji se koriste u ovom receptu su **Person Running** i **Person Walking**. Slika emotikona može se razlikovati u zavisnosti od vašeg OS-a. Slike predstavljene u ovom receptu su emotikoni za Apple OS.

```
watch(() => {
  const street = Array(maxRoadLength).fill('_');
  const marathonEl = document.getElementById('marathon');
  street[competitor.position] = (competitor.position % 2 === 1)
    ? '🏃'
    : '🚶';
  marathonEl.innerHTML = '';
  marathonEl.innerHTML = street.reverse().join('');
});
```



8. Kreirajte funkciju `setInterval`, tako što ćete proslediti anonimnu funkciju kao argument. Unutar funkcije definišite `competitor.position` kao mod funkciju, tako što ćete proslediti `concurrent.position` i `concurrent.speed` kao prvi argument, a `maxRoadLength` kao drugi argument:

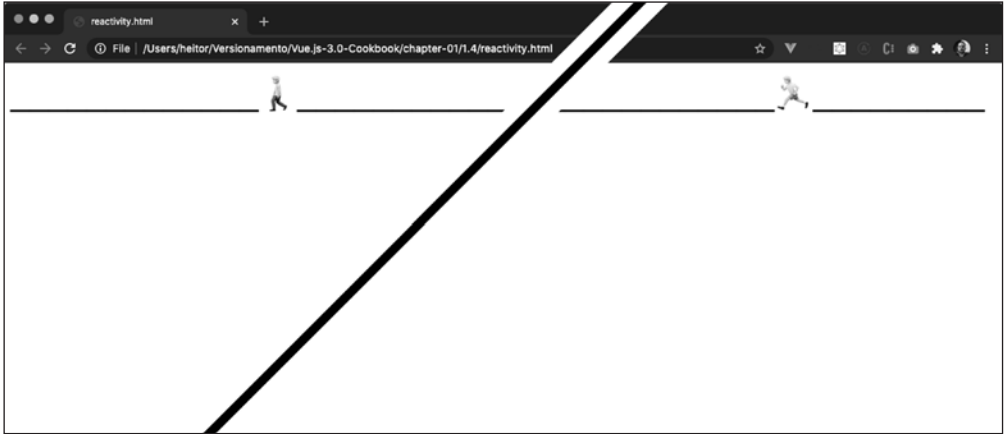
```
setInterval(() => {
  competitor.position = mod(competitor.position + competitor.speed,
    maxRoadLength)
}, 100);
```

Kako to funkcioniše...

Zahvaljujući izloženim API-ima `reactive` i `watch` iz Vuea, uspeli smo da kreiramo aplikaciju pomoću reaktivnosti koja je prisutna u Vue radnom okviru, ali bez upotrebe Vue aplikacije.

Prvo smo kreirali reaktivni objekat `competitor` koji funkcioniše na isti način kao Vue svojstvo `data`. Zatim smo kreirali funkciju `watch` koja funkcioniše na isti način kao i svojstvo `watch`, ali se koristi kao anonimna funkcija. U funkciji `watch` kreirali smo put za takmičara i jednostavnu animaciju pomoću dva različita emotikona koje smo menjali na osnovu položaja na putu, tako da ova funkcija oponaša animaciju na ekranu.

Na kraju, odštampali smo trenutnog trkača na ekranu i kreirali funkciju `setInterval` na svakih 100 ms da bismo promenili položaj takmičara na putu.



Kreiranje komponente pomoću Composition API-a

Composition API je novi način pisanja Vue komponenata, koji je zasnovan na upotrebi funkcija za sastavljanje komponente i čini organizaciju i ponovnu upotrebljivost koda boljom.

Ovaj metod je inspirisan React Hooksom i uvodi tehniku kreiranja specijalne funkcije za sastavljanje aplikacija koje se mogu deliti, a da, pri tom, ne treba da se nalaze u Vue aplikaciji zbog upotrebe izloženih Vue API-a.

U ovom receptu ćete naučiti kako da kreirate spoljnu funkciju koja preuzima geolokaciju korisnika i prikazuje te podatke na ekranu pomoću Composition API-a.

Kako to uraditi...

Ovde ćemo kreirati komponentu pomoću Composition API-a, koja će preuzeti GPS položaj korisnika i prikazaće te informacije na ekranu:

1. Pomoću osnovnog primera iz odeljka „Kreiranje osnovne datoteke“ kreirajte novu datoteku koja se zove `component.html`, a zatim je otvorite.
2. U praznom HTML elementu `<script>` kreirajte konstante funkcija koje će se koristiti pomoću metoda destrukuiranja objekata, tako što ćete pozvati metode `createApp`, `defineComponent`, `setup`, `ref`, `onMount` i `onUnmount` iz globalne konstante `Vue`:

```
const {
  createApp,
  defineComponent,
  setup,
  ref,
  onMounted,
  onUnmounted,
} = Vue;
```

3. Kreirajte funkciju `fetchLocation` i unutar nje kreirajte promenljivu `let` koja se zove `watcher`. Zatim, kreirajte konstantu `geoLocation` i definišite je kao `navigator.geolocation`. Nakon toga, kreirajte konstantu `gpsTime` i definišite je kao funkciju `ref`, tako što ćete proslediti funkciju `Date.now()` kao argument. Na kraju, kreirajte konstantu koja se zove `coordinates` i definišite je kao funkciju `ref`, tako što ćete proslediti JavaScript objekat kao argument pomoću svojstava `accuracy`, `latitude`, `longitude`, `altitude`, `altitudeAccuracy`, `heading` i `speed`, koja su definisana kao 0:

```
function fetchLocation() {
  let watcher;
  const geoLocation = navigator.geolocation;
  const gpsTime = ref(Date.now());
  const coordinates = ref({
    accuracy: 0,
    latitude: 0,
    longitude: 0,
    altitude: 0,
    altitudeAccuracy: 0,
    heading: 0,
    speed: 0,
  });
}
```

4. Zatim, unutar funkcije `fetchLocation`, nakon kreiranja konstanti, kreirajte funkciju `setPosition` pomoću parametra `payload`. Unutar funkcije definišite `gpsTime.value` kao argument `payload.timestamp`, a `coordinates.value` kao argument `payload.coords`:

```
function setPosition(payload) {
  gpsTime.value = payload.timestamp
  coordinates.value = payload.coords
}
```

5. Nakon kreiranja funkcije `setPosition`, pozovite funkciju `onMount`, tako što ćete proslediti anonimnu funkciju kao argument. Unutar funkcije proverite da li pregledač ima dostupan API `geoLocation` i definišite `watcher` kao funkciju `geoLocation.watchPosition`, tako što ćete proslediti funkciju `setPosition` kao argument:

```
onMounted(() => {
  if (geoLocation) watcher =
    geoLocation.watchPosition(setPosition);
});
```

6. Nakon pozivanja funkcije `onMounted`, kreirajte funkciju `onUnmount`, koja prosleđuje anonimnu funkciju kao argument. Unutar funkcije proverite da li je definisan `watcher`, a zatim izvršite funkciju `geoLocation.clearWatch`, tako što ćete proslediti `watcher` kao argument:

```
onUnmounted(() => {
  if (watcher) geoLocation.clearWatch(watcher);
});
```

7. Na kraju, u funkciji `fetchLocation` vratite JavaScript objekat kao rezultat i, kao što svojstva/vrednosti definišu, prosledite konstante `coordinates` i `gpsTime`:

```
return {
  coordinates,
  gpsTime,
};
```

8. Kreirajte konstantu `appComponent` i definišite je kao funkciju `defineComponent`, tako što ćete da prosledite JavaScript objekat pomoću svojstava `setup` i `template` kao argumenata:

```
const appComponent = defineComponent({
  setup() {},
  template: ``
});
```

9. U funkciji `setup` kreirajte konstantu koja predstavlja destrukuiranje objekta pomoću svojstava `coordinates` i `gpsTime` funkcije `fetchLocation`:

```
setup() {
  const {
    coordinates,
    gpsTime,
  } = fetchLocation();
}
```

10. Unutar funkcije `setup` kreirajte drugu konstantu koja se naziva `formatOptions` i definišite je kao JavaScript objekat pomoću svojstava `year`, `month`, `day`, `hour` i `minute` kao `'numeric'`. Zatim, definišite svojstvo `hour12` kao `true`:

```
const formatOptions = {
  year: 'numeric',
  month: 'numeric',
  day: 'numeric',
  hour: 'numeric',
  minute: 'numeric',
  hour12: true,
};
```

11. Nakon kreiranja konstante `formatOptions`, kreirajte konstantu koja se zove `formatDate` i definišite je kao funkciju koja prima parametar `date`. Zatim, vratite novu funkciju `Intl.DateTimeFormat` kao rezultat, tako što ćete proslediti `navigator.language` kao prvi, a konstantu `formatOption` kao drugi argument. Izradite lanac prototipova funkcije `format`, tako što ćete proslediti parametar `date`:

```
const formatDate = (date) => (new
  Intl.DateTimeFormat(navigator.language,
    formatOptions).format(date));
```

12. Konačno, na kraju funkcije `setup` vratite JavaScript objekat kao rezultat pomoću svojstava koja su definisana kao konstante `coordinates`, `gpsTime` i `formatDate`:

```
return {
  coordinates,
  gpsTime,
  formatDate
};
```

13. U svojstvu `template` uradite sledeće:

- Kreirajte HTML element `h1` pomoću teksta `"My Geo Position at {{formatDate(new Date(gpsTime)) }}"`.
- Kreirajte HTML element `ul` i dodajte tri HTML elementa `li` kao podređene elemente.
- U prvi podređeni element dodajte tekst `"Latitude: {{coordinates.latitude }}"`.
- U drugi podređeni element dodajte tekst `"Longitude: {{coordinates.longitude }}"`.
- U treći podređeni element dodajte tekst `„Altitude: {{coordinates.altitude }}"`:

```
template: `
  <h1>My Geo Position at {{formatDate(new
                                Date(gpsTime))}}</h1>
  <ul>
    <li>Latitude: {{ coordinates.latitude }}</li>
    <li>Longitude: {{ coordinates.longitude }}</li>
    <li>Altitude: {{ coordinates.altitude }}</
    li>
  </ul>
`
```

14. Na kraju, pozovite funkciju `createApp`, tako što ćete proslediti konstantu `appComponent` kao argument. Zatim, izradite lanac prototipova funkcije `mount` i, kao argument funkcije, dodajte atribut `id` HTML elementa `div("#app")`:

```
createApp(appComponent)
  .mount('#app');
```

Kako to funkcioniše...

U ovom receptu smo prvo uvezli izložene API-e `createApp`, `defineComponent`, `setup`, `ref`, `onMount` i `onUnmount` kao konstante, koje ćemo koristiti za kreiranje komponente. Zatim smo kreirali funkciju `fetchLocation` koja je odgovorna za dobijanje korisničkih podataka o geolokaciji i za njihovo vraćanje kao reaktivnih podataka koji se mogu automatski ažurirati kada korisnik menja njihovu lokaciju.

Preuzimanje korisničkih GPS pozicija bilo je moguće zahvaljujući API-u `navigator.geolocation` koji je prisutan u savremenim pregledačima, a koji mogu da preuzmu trenutni GPS položaj korisnika. Pomoću ovih podataka koje obezbeđuje pregledač uspeali smo da definišemo promenljive koje su kreirane pomoću API-a `ref`.

Komponentu smo kreirali pomoću funkcije `setup` deklaracije objekta Vue, tako da renderovanje „zna“ da koristimo novi Composition API kao metod kreiranja komponente. Unutar funkcije `setup` uvezli smo dinamičke promenljive funkcije `fetchLocation` i kreirali smo metod koji formatira datum za upotrebu kao filter u šablonu.

Zatim smo vratili uvezene promenljive i filter, tako da se mogu koristiti u odeljku šablona. U tom odeljku kreirali smo naslov, dodajući vreme poslednjeg GPS položaja, koristili smo filter za njegovo formatiranje i kreirali smo listu geografske širine, dužine i nadmorske visine korisnika.

Na kraju, kreirali smo aplikaciju pomoću izloženog API-a `createApp` i montirali smoVue aplikaciju.

Takođe pogledajte

Više informacija o API-u `Navigator.geolocation` možete pronaći na adresi <https://developer.mozilla.org/en-US/docs/Web/API/Navigator/geolocation>.

Više informacija o API-u `Intl.DateTimeFormat` možete pronaći na adresi https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Intl/DateTimeFormat.