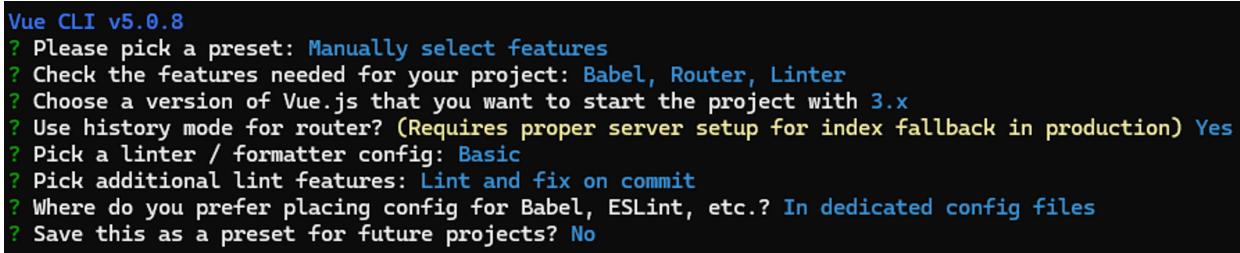


Vue.js

- instalira se node.js I nakon toga se kuca komanda: **npm install -g @vue/cli**
 - ona je na vezbama pogresila jer nije stavila -g, a to mora da se napise (da bi se videlo globalno)
- njoj je radilo pokretanje iz vs terminala, ali ako to ne radi onda je resenje da se pokrene iz komandne linije (cmd, a ne powershell)
 - ako pise kao da je dobro instalirano ali kada se kuca neka vue komanda kaze da vue ne postoji onda treba da se instalira kao admin, na sledeci nacin:
 - i. prvo se napise npm uninstall -g @vue.cli
 - ii. onda se pre otvaranja cmd-a klikne desni klik na njega I odabere Run as Administrator
 - iii. u tom cmd-u se napise opet komanda za instaliranje I ovako treba da radi
- meni nije radilo nakon ovoga kada se bilo koja vue komanda kuca iz vs terminala, nego sve mora iz cmd (za ovo ne mora da bude admin)

Kreiranje vue aplikacije:

- **vue create myapp**
 - odaberemo opciju Manually select features, pa izaberemo Babel, Router I Linter/Formatter (plus ako izadje I opcija Choose Vue version)

 - treba mu neko vreme da sve instalira za aplikaciju
- nakon pokretanja pokrenemo komandu **cd myapp**, pa zatim **npm run serve** gde ce izaci adresa u browseru na kojoj mozemo da pristupimo nasoj aplikaciji (localhost:8080)

Vue aplikacija:

main.js fajl:

```
import { createApp } from 'vue'  
// importovanje funkcije createApp iz vue bibliotekе  
import App from './App.vue'  
// u fajlu App.vue je automatski napravljena komponenta App koja se ovde importuje  
import router from './router'  
// isto kao I za App, router je komponenta koja je napravljena u './router' I ovde se importuje  
  
createApp(App).use(router).mount('#app')  
// createApp(App) stvara instancu aplikacije cija je glavna komponenta App.vue  
// .mount('#app') montira Vue aplikaciju na HTML element s id-jem app, I to je mesto gde ce Vue  
aplikacija renderovati svoj sadrzaj  
// taj element je div koji je u fajlu index.html
```

- ovo se dobija pri pokretanju aplikacije I ne moramo da ga menjamo

App.vue fajl:

```
<template>
  <nav>
    <router-link to="/">Home</router-link> |
    <router-link to="/about">About</router-link>
  </nav>
  <router-view/>
</template>
```

- ovo je komponenta koja koristi Vue ruter za upravljanje rutama
- **router-view** je ugradjena komponenta koja se koristi za dinamicko prikazivanje komponenti u zavisnosti od trenutne rute aplikacije
- ovo rutiranje sluzi da se sadrzaj stranice menja bez potrebe da se ponovo ucitava stranica
 - sve je jedna stranica -> **index.html** a onda se u njen div sa id-jem app ubacuje i menja sadrzaj u zavisnosti od rute
- u deo router-view se ubacuje trenutni sadrzaj putanje
 - dakle, ako je putanja "/" sadrzaj Home stranice se ugradjuje u router-view i tako prikazuje

router/index.js fajl:

```
import { createRouter, createWebHistory } from 'vue-router'
import HomeView from '../views/HomeView.vue'

const routes = [
  {
    path: '/',
    name: 'home',
    component: HomeView
  },
  {
    path: '/about',
    name: 'about',
    // route level code-splitting
    // this generates a separate chunk (about.[hash].js) for this route
    // which is lazy-loaded when the route is visited.
    component: () => import(/* webpackChunkName: "about" */ '../views/AboutView.vue')
  }
]

const router = createRouter({
  history: createWebHistory(process.env.BASE_URL),
  routes
})

export default router
```

- u ovom fajlu se definise rutiranje, tako da je svaki url mapiran na odredjenu komponentu
- HomeView je komponenta napravljena automatski unutar views foldera:

```
<template>
  <div class="home">
    
    <HelloWorld msg="Welcome to Your Vue.js App"/>
  </div>
</template>

<script>
// @ is an alias to /src
import HelloWorld from '@/components/HelloWorld.vue'

export default {
  name: 'HomeView',
  components: {
    HelloWorld
  }
}
</script>
```

- **export default** sluzi da se od ovog koda **napravi komponenta I da se exportuje** tako da drugi fajlovi mogu da ga uvoze
- **components** deo sluzi da se navede koje komponente sadrzi I koristi komponenta koja se pravi
- HelloWorld je komponenta koja je napravljena automatski u folderu components I ovde je uvezena komandom **import HelloWorld**
- ekstenzija za nove komponente koje pravimo je **.vue**
- Bootstrap link se dodaje u head deo **index.html stranice**

Izmenjen HomeView.vue:

```
<template>
  <div class="home">
    <h1 :class='h1class'>Welcome to our bookstore {{bookStoreName}}</h1>
    <br>
    <button class="btn btn-success">Sign in</button>
  </div>
</template>

<style>
  .extraClass{
    color: purple
  }
</style>

<script>
  export default {
    name: 'Home',
    data(){
      return{
        bookStoreName: "Booka",
        h1class: 'extraClass'
      }
    }
  }
</script>
```

- sav sadrzaj za stranicu se pise unutar **template** tagova
- **data()** metoda sluzi za definisanje podataka koji ce biti korisceni unutar komponente
 - ova metoda vraca objekat koji sadrzi podatke koje Vue treba reaktivno da prati I dinamicki menja
 - npr. ako imamo bookStoreName promenljivu, kada je promenimo unutar data() dela svuda na stranici gde je koriscen ce se automatski azurirati
 - oznacili smo h1class kao promenljivu kojoj smo dodelili definisani 'extraClass', a ona se koristi ovako:
 - <h1 :class='h1class'>
 - mora da se stavi : pre class

Navigation.vue:

- ubacili smo samo nav deo iz App.vue fajla I stilizovali ga
- export:

```
<script>
  export default {
    name: 'Navigation'
  }
</script>
```

- u App.vue smo ga dodali na sledeci nacin:

```
<template>
  <Navigation></Navigation>
  <router-view/>
</template>
```

- koristimo ime komponente **kao tag**

- **BITNO:** sve komponente koje koriste druge komponente moraju da se izvoze pomocu export funkcije, zato dodajemo export u App.vue

```
<script>
  import Navigation from '@/components/Navigation.vue'

  export default {
    name: 'App',
    components: {
      Navigation
    }
  }
</script>
```

Book.vue:

```
<template>
  <div id='book'>
    <b>Title:</b> {{mybook.title}}
    <hr>
    <i>
      Author: {{mybook.author}}
      Year: {{mybook.publishYear}}
      Desc: {{mybook.description}} </i>
    </div>

  </template>

  <script>
    export default{
      name:'Book',
      props: [
        'mybook'
      ]
    }
  </script>
```

- **props** je kljuc u eksportovanju u koji dodajemo properties komponente koju pravimo, koje cemo kasnije koristiti kao imena atributa tog taga

- unutar **{ }** zagrada se dohvataju vrednosti koje su exportovane u props delu

ShowBooks.vue:

- pravimo js fajl u kome cemo definisati niz knjiga koje treba da prikaze nasa stranica, ali da bi taj niz mogao da se koristi u vue komponentama, I za njega mora da se uradi export

```
> const allBooks=[ ...
]

export default allBooks;
```

- import se radi kao inace: import allBooks from '../data/books.js'

- **BITNO:** ako ovako definisemo niz I zelimo da ga eksportujemo, mora da se oznaci sa **const**

- kod ShowBooks.vue:

```
<template>
  <div id='showbooks'>
    <table>
      <tr v-for='book in books' :key='book.id'>
        <td>
          <div>{{book.title}}</div>
          <Book :mybook="book"></Book>
        </td>
      </tr>
    </table>
  </div>
</template>

<script>
  import Book from './Book.vue'
  import allBooks from '../data/books.js'
  export default{
    name:'ShowBooks',
    components:{
      Book
    },
    data(){
      return{
        books:allBooks
      }
    }
  }
</script>
```

- **BITNO:**

- komponenti dajemo ime 'ShowBooks' prilikom eksportovanja, i bitno je da se **za definisanje imena koriste navodnici**, a za spisak komponenata nisu potrebni navodnici -> components:{ **Book** }

- kod taga Book koristimo **props** atribut *mybook* koji smo definisali prilikom pravljenja komponente Book, i prosledujemo mu objekat "book" (mora pod navodnicima), koji se onda koristi u samoj komponenti za dobijanje polja title, publishYear itd.
- unutar data dela vracamo promenljive koje koristimo u komponenti
- **v-for** je for petlja koja označava da se za svaki objekat book u nizu books koji je prosledjen pravi poseban <tr> element
 - :key je ugradjeni property
- u HomeView.vue komponentu smo dodali da se importuje ova komponenta, i da se navodi u spisku components:

```
<script>
  import ShowBooks from '../components>ShowBooks.vue'
  export default {
    name: 'Home',
    components: { ShowBooks },
```

- ovu komponentu koristimo kao tag <ShowBooks></ShowBooks> na kraju diva "home"
- ako se koristi **<style scoped>** to znači da se taj stil primenjuje samo na trenutnu komponentu

BookDetails.vue:

- da bi se omogucio pristup stranici za svaku knjigu klikom na njen tag u tabeli, onda mora da se oko tog taga stavi tag **<routerr-link>**

```
<tr v-for='book in books' :key='book.id'>
  <td>
    <routerr-link :to='/bookdetails/+book.id'>
      <Book :mybook="book"></Book>
    </routerr-link>
  </td>
</tr>
```

- **:to** označava na koju lokaciju treba ruter da predje

- tu rutu dodajemo u spisak ruta (u index.js fajlu):

```
{
  path: '/bookdetails/:id',
  name: 'BookDetails',
  component: BookDetails
}
```

- :id** označava da će taj parametar putanje biti prosledjen (da bismo posle mogli da ga koristimo)

- BookDetails.vue kod:

```
<template>
  <div>
    <h3>{{book.title}}</h3><hr/>
    
  </div>
</template>

<script>
  import allBooks from "../data/books.js"
  export default{
    name:'BookDetails',
    created(){
      var bookId=Number(this.$route.params.id)
      this.book = this.books.find(book=>book.id==bookId)
    },
    data(){
      return{
        books:allBooks,
        book:{}
      }
    }
  }
</script>
```

- ukoliko želimo da se putanja do neke slike ucitava dinamicki te slike **moramo da dodamo u public folder**, a u ovom slučaju smo u tom public folderu kreirali I folder **images** iz kog citamo sve slike -> direkno se src slike cita iz public-a, ne mora da se stavlja public u putanju
- created()** je dogadjaj koji se okida kada je komponenta kreirana
 - definisali smo da se tada pronadje id preko putanje na kojoj smo: **this.\$route.params.id**, zato sto smo id prosledili kao parametar te putanje
 - this.books.find(...) pronalazi knjigu sa datim id-jem u nizu books koji je prosledjen u data() podacima
 - this** se koristi za promenljive koje su definisane u **data()**

Definisanje metode:

- u HomeView.vue export delu smo dodali export metode:

```
methods:{
  hello(name){
    alert(name)
  }
}
```

- tako prosledjena metoda može da se zove na sledeći nacin:

```
<button class="btn btn-success" @click="hello('pera')">Sign in</button>
  @click=...
```

Vue-final-modal:

- ovo je vue biblioteka koja omogucava kreiranje i upravljanje modalnim prozorima (dijaloga, obavestenja i sl.)
- instalacija: **npm install vue-final-modal@next**
- import: `import {VueFinalModal, $vfm} from 'vue-final-modal'`
 - `$vfm` je globalni vue-final-modal objekat koji omogucava direktnu interakciju sa modalnim prozorima -> pomocu njega mogu da se otvore, zatvore i dobiju informacije o trenutnom stanju
 - mora da se doda u i components:

```
components:{ ShowBooks, VueFinalModal },
```

- sav sledeci kod napisan je u HomeView.vue fajlu:
- dodat je modalni prozor za sign in, tako da se na pritisak dugmeta *Sign in* prikaze forma:

```
<button class="btn btn-success" @click="openLoginModal()">Sign in</button>
<hr>
<ShowBooks></ShowBooks>
<div>
  <vue-final-modal v-model="showModal" name="loginModal" classes="modal-container">
    <table>
      <tr>
        <td style="text-align:right">
          <button class="modal__close" @click="showModal = false">
            X
          </button>
        </td>
      </tr>
    </table>

    <table>
      <tr>
        <td>Username:</td>
        <td><input type='text' name='username' v-model="username"></td>
      </tr>
      <tr>
        <td>Password:</td>
        <td><input type='password' name='password' v-model="password"></td>
      </tr>
      <tr>
        <td colspan="2"><button @click="confirm()">Confirm</button></td>
      </tr>
    </table>
  </vue-final-modal>
</div>
```

- metodu `openLoginModal` smo definisali u methods delu kao:

```
openLoginModal(){
  $vfm.show('loginModal')
},
```

- iskoristili smo objekat `$vfm` koji smo importovali, i pomocu metode `show` prikazali modal za zadatim imenom (atribut `name='loginModal'`)
 - postoji i metoda `hide` sa suprotnim efektom
- **BITNO:** kada se zadaje koja metoda se poziva u `@click`, mora da se navede **poziv metode**, tj. moraju da se stave zagrade `()`, a ne samo ime metode
- metodu `confirm` smo definisali u methods delu kao:

```
confirm(){
  var user=allUsers.find(user=>user.username==this.username && user.password==this.password)
  if(user)alert(user.firstname)
  else alert("error")
}
```

- napravili smo `allUsers` isto kao i `allBooks` niz i onda smo ga importovali
- koristimo metodu `find` da u nizu recnika nadjemo odredjenog usera

- u data delu smo definisali promenljive this.username I this.password, koje su pri ucitavanju prazne:

```
data(){
  return{
    bookStoreName: "Booka",
    h1class: 'extraClass',
    showModal: false,
    username:"",
    password:""
  }
},
```

- da bismo input tagove koje smo napravili za ime I lozinku vezali za ove promenljive **iz data dela** koristimo atribut **v-model** koji postavljamo da bude jednak imenu promenljive za koji ga vezujemo:

<input type='text' name='username' v-model="username">

- ovo vezuje input "username" sa promenljivom this.username

- promenljiva **showModal** je reaktivna promenljiva koja se koristi za kontrolu vidljivosti modalnog prozora u Vue.js aplikaciji

```
<vue-final-modal v-model="showModal" name="loginModal" classes="modal-container"
content-class="modal-content">
  <table>
```

- da bi se koristila kao reaktivna promenljiva (da bi je vue model prepoznao), mora da se stavi u tag: <vue-final-modal **v-model="showModal"** >
- pri ucitavanju smo je postavili na **false**, sto znaci da se ne prikazuje, sve do pritiska dugmeta *Sign in* I pozivanja metode openLoginModal
 - na pritisak dugmeta *Sign in* smo mogli da stavimo da se izvrsava i: "showModal=true" I imalo bi isti efekat kao I \$vm.show('loginModal')
- pritiskom na dugme X koje smo definisali za zatvaranje forme se ova promenljiva postavlja na false:

```
<button class="modal__close" @click="showModal = false">
  x
</button>
```

- dugme X koje smo dodali smo prekopirali sa vue-final-modal sajta, kao I ceo <style> tag za centriranje forme

Ulogovanje korisnika prema tipu (user/admin):

- u HomeView.vue dodajemo:

```
created(){
  if(localStorage.getItem("allUsers")==null){
    localStorage.setItem("allUsers",JSON.stringify(allUsers))
  }
}
```

- prilikom ucitavanja stranice (*created*) u local storage ubacujemo sve korisnike da bismo mogli kasnije iz njega da citamo
- **stringify** pretvara recnik u string

- promjenjena confirm metoda:

```
confirm(){
  // var user=allUsers.find(user=>user.username==this.username && user.password==this.password)
  var users=JSON.parse(localStorage.getItem("allUsers"))
  var user=users.find(user=>user.username==this.username && user.password==this.password)
  if(user){
    if(user.type==0){
      this.$router.push("user");
    }else{
      this.$router.push("admin");
    }
  }else{
    alert("Error")
  }
}
```

- umesto iz recnika allUsers, cita se iz local storage-a

- **parse** pretvara string koji je u JSON formatu u JSON recnik
- u odnosu na to koji je tip ulogovan prebacujemo se na stranicu za usera ili za admina, koje smo napravili u views folderu: User.vue I Admin.vue, I onda ih importovali kao komponente
 - u router deo (index.js fajl) moramo da dodamo te 2 nove putanje: ('/user' I '/admin')
 - **this.\$router.push("user")** prebacuje rutu na onu u zagradama

Admin.vue:

- na admin stranici pravimo formu za dodavanje novog user-a

```
<template>
  <div class="admin">
    <h3>Add user</h3>
    <div class="addUserTable">
      <table>
        <tr>
          <td> Firstname: </td>
          <td> <input type="text" name="firstname" v-model="firstname"></td>
        </tr>
        <tr>
          <td> Lastname: </td>
          <td> <input type="text" name="lastname" v-model="lastname"> </td>
        </tr>
        <tr>
          <td> Username: </td>
          <td> <input type="text" name="username" v-model="username"> </td>
        </tr>
        <tr>
          <td> Password: </td>
          <td> <input type="text" name="password" v-model="password"> </td>
        </tr>
        <tr>
          <td colspan="2">
            <button @click="addUser()"> Add user </button>
          </td>
        </tr>
      </table>
    </div>
  </div>
</template>
```

- export:

```
export default {
  name: "Admin",
  data(){
    return{
      firstname:"",
      lastname:"",
      username:"",
      myUsers:[]
    }
  },
  methods:{
    addUser(){
      this.myUsers.push({
        "firstname":this.firstname,
        "lastname":this.lastname,
        "username":this.username,"password":this.password,"type":0})
      localStorage.setItem("allUsers",JSON.stringify(this.myUsers))
    }
  },
  created(){
    this.myUsers=JSON.parse(localStorage.getItem("allUsers"))
  }
}
</script>
```

- kada se ucitava stranica u polje myUsers treba da ucitamo korisnike iz local storage-a
- pritiskom na dugme Add user poziva se metoda addUser, koja u listu myUsers dodaje korisnika sa upisanim podacima, pa se azurira local storage
 - v-model atribut povezuje input polja sa poljima u data delu, tako da se this.firstname, this.lastname I this.username popunjavaju cim se popuni forma

User.vue:

- na stranici usera treba da se ispise spisak svih knjiga objavljenih u prethodnih 10 dana

```
<template>
  <div>
    <ul>
      <li v-for='book in myBooks' :key='book.id'>
        {{book.title}}
      </li>
    </ul>
  </div>
</template>

<script>

  import allBooks from "../data/books.js"
  export default {
    name: "User",
    data(){
      return{
        myBooks:[]
      }
    },
    created(){
      for(var i=0;i<allBooks.length;i++){
        var today=new Date();
        var datum=allBooks[i].publishYear.split("-")

        // meseci se indeksiraju od 0
        var dan=parseInt(datum[0])
        var mesec=parseInt(datum[1])-1
        var god=parseInt(datum[2])
        var bookDate=new Date(god,mesec,dan)

        // razlika u milisekundama
        var diff=today.getTime()-bookDate.getTime();
        diff=diff/(1000*60*60*24)
        if(diff<10){
          this.myBooks.push(allBooks[i])
        }
      }
    }
  }
</script>
```

- importujemo niz allBooks
- u myBooks niz u data delu cemo upisivati sve knjige koje zadovoljavaju gorepomenuti uslov
- prilikom ucitavanja stranice prolazimo kroz niz allBooks i proveravamo datum sa danasnjim:
 - new Date()** pravi objekat trenutnog datuma
 - od godine, meseca i dana objavlivanja knjige pravimo objekat datuma
 - BITNO:**
 - meseci se indeksiraju od 0, pa se zato oduzima 1
 - u konstruktor se redom prosledjuju: godina, mesec, dan
 - racunanje razlike:

```
var diff=today.getTime()-bookDate.getTime();
diff=diff/(1000*60*60*24)
```

 - getTime()** vraca vrednost datuma u milisekundama, pa da bismo dobili koliko je dana izmedju nekog datuma delimo tu razliku sa:
1000 (milisekundi u sekundi) * 60 (sekunde u minuti) * 60 (minuti u satu) * 24 (sata u danu)

Cas 11.2 - rok:

- ako se desi ova najgluplja greska to znaci da umesto da nazovemo komponente Kupac, Pocetna I Zaposleni, treba da ih nazovemo tako da sadrze **vise reci** npr. KupacView, PocetnaView I ZaposleniView:

```
You may use special comments to disable some warnings.  
Use // eslint-disable-next-line to ignore the next line.  
Use /* eslint-disable */ to ignore all warnings in a file.  
ERROR in [eslint]  
C:\Users\ANA\Desktop\Faks\6. semestar\Veb dizajn\Cas 11 - rok\myapp\src\views\Kupac.vue  
9:14  error  Component name "Kupac" should always be multi-word  vue/multi-word-component-names  
  
C:\Users\ANA\Desktop\Faks\6. semestar\Veb dizajn\Cas 11 - rok\myapp\src\views\Pocetna.vue  
31:14  error  Component name "Pocetna" should always be multi-word  vue/multi-word-component-names  
  
C:\Users\ANA\Desktop\Faks\6. semestar\Veb dizajn\Cas 11 - rok\myapp\src\views\Zaposleni.vue  
9:14  error  Component name "Zaposleni" should always be multi-word  vue/multi-word-component-names  
  
✖3 problems (3 errors, 0 warnings)
```

- drugo resenje je da u ".eslintrc.js" fajl se u modules.export doda:

```
rules: {  
  'vue/multi-word-component-names': 'off',  
}
```

- ako se desi:

```
[eslint]  
C:\Users\ANA\Desktop\Faks\6. semestar\Veb dizajn\Cas 11 - rok\myapp\src\views\Pocetna.vue  
101:64  error  'myUsers' is not defined  no-undef  
  
✖1 problem (1 error, 0 warnings)
```

- to znaci da nije stavljen this.myUsers, nego direktno myUsers (ili ukoliko je bila lokalna promenljiva nije definisana sa var ili let)
- da bi se stavilo da je neko input polje **required** mora da se okruzi tagom **<form>**
- bitno je da se uvek kada se dodaje objekat u localStorage koristi metoda JSON.stringify, a kada se uzima JSON.parse

- ako se to negde zaboravi desice se greska:

```
"[object Object]" is not valid JSON  
SyntaxError: "[object Object]" is not valid JSON
```

- koriscenje parametra u metodi:

```
<tr v-for="product of products" :key="product.naziv" :class="product.  
tip=='Visegodisnj'a?'boja':''">  
  <td>{{product.naziv}}</td>  
  <td>{{product.kolicina}}</td>  
  <td>{{product.tip}}</td>  
  <td>  
    <button @click="method(product)">  
      Action  
    </button>  
  </td>  
</tr>
```

- moze method da se pozove sa parametrom product koji je tekuci product u v-for petlji izlistavanja elemenata
- uklanjanje elementa iz niza:

```
method(p){  
  // alert(p.naziv)  
  let index=this.products.indexOf(p)  
  this.products.splice(index,1)  
  localStorage.setItem("products",JSON.stringify(this.products))  
},
```

- **indexOf** - vraca indeks gde se nalazi zadati element
- **splice** - uklanja toliko elemenata koliko je oznaceno drugim parametrom, pocev od indeksa koji je dat kao prvi parametar

- **BITNO:**
 - ukoliko treba da se doda neka klasa/stil u zavisnosti od prosledjenog podatka moze da se koristi ternarni operator kao sto je napisano gore:


```
<tr v-for="product of products"
      :key="product.naziv" :class="product.tip=='Visegodisnja'? 'boja':''">
```

 - sintaksa: :class="uslov?'vrednost1':'vrednost2'"
 - u ovom primeru vrednost2 je " (prazan string)
 - primer za style:


```
<div :style="username==''? 'display:none': ''">
```
- logout:


```
logout(){
  localStorage.removeItem("loggedIn")
  this.$router.push("/")
}
```

 - cuvamo da je ulogovan u local storage-u u funkciji login, a u funkciji logout ga brišemo
 - **BITNO:**
 - ne moze da se obriše nesto iz local storage-a koriscenjem localStorage.setItem("x",null), jer ga nece postaviti na objekat null vec na string "null", pa provera: if(localStorage.getItem("x")==null) nece biti dobra
 - mora da se koristi **removeItem**

Css pre-processors:

- prilikom pokretanja aplikacije treba da se selektuje kao feature

```
? Please pick a preset: Manually select features
? Check the features needed for your project: Choose Vue version, Babel, Router, CSS Pre-processors, Linter
? Choose a version of Vue.js that you want to start the project with 3.x
? Use history mode for router? (Requires proper server setup for index fallback in production) Yes
? Pick a CSS pre-processor (PostCSS, Autoprefixer and CSS Modules are supported by default): Less
? Pick a linter / formatter config: Basic
? Pick additional lint features: Lint and fix on commit
? Where do you prefer placing config for Babel, ESLint, etc.? In dedicated config files
? Save this as a preset for future projects? No
```

- primer koriscenja:

```
<template>
  <div class="about">
    <h1>This is an about page</h1>
    <button class='ok'>OK</button>
    <button class='cancel'>CANCEL</button>
  </div>
</template>

<style lang="less" scoped>
  @bg: #lightgreen;

  h1{
    background-color: @bg;
    color: multiply(#red, #lightgreen)
  }

  .ok{
    font-weight: bold;
  }

  .cancel:extend(.ok){
    font-style: italic;
  }
</style>
```

- u style treba da se ubaci `lang="less"`

- sa @bg smo definisali promenljivu koju mozemo da koristimo u ostatku koda
- mogu da se pozivaju ugradjene funkcije unutar style dela, kao npr. multyply za boje (mesanje 2 boje)
- ovaj feature omogucava nasledjivanje stilova: **extend**(.ok)

Deployment:

- komanda: npm run build
 - kreira se dist folder
- komanda: serve -s dist
 - aplikacija se otvara na portu 3000