Day 1

* vscode da gözükmesini istemediğimiz dosya için ayarlarda exclude ara ve add item ile ekle
* cli klasörü oluşturmak için vue create dosya adı
* bootstrap linki public içindeki index.html deki head etiketinin içine yapıştır

    <link

      href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css"

      rel="stylesheet"

      integrity="sha384-GLhlTQ8iRABdZLl6O3oVMWSktQOp6b7In1Zl3/Jr59b6EGGoI1aFkw7cmDA6j6gD"

      crossorigin="anonymous"

    />

* console.log kısayolu clo yazıp enter a basmak clg yi dene
* alt ve click yapınca bir imleç daha ekleniyor
* tasarımlar bootstrap ile yapılacak bootstrap dersleri izlenecek
* koz yazarken auto completede özellikle class= ile id= e dikkat et
* id= olacak vue nun çalışması için div#app yazarsak id="app" olan bir div açar
* head elementinin içinde <script src="https://unpkg.com/vue@next"></script>

vue yu görmesi için

* {{title}} string interplution demek oluyor text contentinin olduğu yerlerde çalışır

bunu elementin içindeki property de yazdırmak için v-bind:href="url" şeklinde

yapıp kullanabiliriz yani string interplution a çevirmek gibi

* json object verilerini kullanmak için yani süslüler içindeki verileri attrubute de kullanmak için attribute başına üstteki gibi v-bind:target="eduflow.target"

{{eduflow.title}} gibi v-bind yerine : koymak yeterli

* butonlarda v-on:click kullanıyoruz vue daki methods u kullanmak için v-on:click="changeTitle"
* herhangi bir nesneye mesela div'e click özelliği vermek için @click="" fonksiyon içinde fonksiyon çalıştırmak için fonksiyonun içinde this.changeTitle() yapmak yeterli
* v-on:input ile onChange eşdeğer @input da yazılabilir input içerisine ama onChange yani updateValue fonksiyonunu vue kısmında yazıyoruz html tarafında değil input dan target.value alabiliriz

updateValue(event){

console.log(event.target.value);

}

* yazılan veriyi almada fonksiyondan kurtulmak için @input="fullName = $event.target.value" yapıp methods içindeki fonksiyondan kurtulmuş oluruz
* v-model="fullName" yaparsak yukardaki işlemdende kurtulmuş oluruz yazımız anlık olarak değişir
* virtual dom değişikliği algılar ve dom sonra update yapar sayfayı o yüzden sayfadaki fonksiyonların hepsi çalışır yani iki counter1 butonuna basınca counter1 ve counter2 çalıştı yazar ama counter1 çalıştı yazması isteniyorsa o zaman methods:{} tanımlar gibi computed:{} tanımlayıp içine değişiklik olabilen fonksiyonlarımızı atacaz ama computed lar değişken gibidir geriye return döndürmesi gerekir yani returnlu fonksiyonlar olacak
* yeni vue da computed içerisindeki fonksiyonlarda html kısmında () kullanılmaz artık computed property diye kavram var yani method değiller
* en başta virtual dom değişikliği algılıyor ve ilk açılışta tüm fonksiyonlar çalışıyor
* created hooku flutter daki initState diyebiliriz
* htmlde vue ile elementlerin görünür olma durumu v-if="showContainer" true false ayarlanır butonun click eventine göre p veya div neyse gözükür gözükmez
* v-for ile de döngü verebiliriz element içerisinde v-for="list in todoList"

Day 2

* componentler string template ve single template olarak ikiye ayrılıyor
* componentleri çağırırken /> şeklinde değil normal şekilde çağır
* cli da proje oluşturmak için vue create dosya adı
* npm run serve ile ayağa kaldırıyoruz
* vue uzantılı dosyalar bir component dosyasıdır single file component sfc oluyor yani
* içerisine 3 farklı yapı alır template- script - style
* style.css asset klasörüne yapıştır uygulama genelinde kullanmak için main de import et "@/assets veya /src/assets/
* vue com da style guide ı incele
* bootstrapı da aynı yukardaki gibi import edebiliriz @ işareti src/ demektir ama pluginleri var onları kullanmak daha doğru ilerleyen günlerde görülecek
* oluşturduğumuz componenti dış dünyaya açmak için script tagı içerisinde export default{data(){return{değişkenlerimiz};},} şeklinde ayarlıyoruz buna options api deniyor
* başta linti dahil etmek demek proje içinde bir şeyi çağırdığında kullanmak zorunda olduğumuzu söyleyen bir şey
* componentleri oluştururken dosya uzantısına dikkat .vue şeklinde olacak
* key ve value aynı olan arkadaşları components:{HomeItem: HomeItem} yazmak yerine components:{HomeItem} şeklinde yazabiliriz. Babel bizim yerimize arka tarafta ayarlar
* /> şeklinde kapatmaya self closing deniyor
* global şekilde component tanımlayabiliriz header footer için özellikle her yere import yazmaya gerek yok main de import global registiring, diğer local registiring demek
* html de bir etiketi çoğaltmak için vue da v-for="i in 5":key="i" şeklinde labeli veya li yi 5 tane çoğaltabilir 5 tane li kopyala yapıştır yapmak yerine yazdırma kısmında ayırt edilmesi için {{i}} şeklinde yazacaz

Components

* vue ile componentler arasında iki tür veri iletişimi var biri parent dan child a yani props diğeri child dan parent a yani custom event app.vue ana component yani parent component oluyor
* Component import ederken böyle değil '@/com...' böyle olmalı "@/com..."
* props ile tanımladığımız veriye aynı data içerisnde return içerisinde veri oluşturup kullandığımız gibi kullanıyoruz {{myData}}
* <DataComponent myData="WDKWOKD"/> bu dinamik olarak gönderme olmuyor başına v-bind: koyarsak dinamik olarak göndeririz yani dinamik olarak göndermek derken mesela bir array var onu göndericez o zaman myData="WDKWOKD"/> yazmak yerine v-bind:myData="todoList" şeklinde data içerisinde tanımladığımız değişkeni ve değerini göndermiş oluyoruz
* vue 3 ün en güzel özelliklerinden biri geçiş sebebi bile denilebilir componentlerde bir veriyi köprü komponentlerle aktarmadan yani sürekli constructor kullanmak gibi yerine getxdeki arguments misali vue da provide inject yöntemi var.İilgili veriyi ilgili komponente gönderir arada köprü componentler olmaz
* provide(){return{mydata:this.mydata};} bu mydata değişkenini hangi component kullanacaksa inject:["mydata"] provide edilmiş bilgiyi yazıyoruz aynı props gibi
* ama... async işlerde api den veri gelmesi olsun anlık değişen işlerde sıkıntı oluyor çözümü uydurulmuş bir değişken oluşturmak örnek provideData:{todoList:["",""]} şeklinde bu sefer todoList yani static olan array değişkenini sarmalayınca bu sefer sorun çözülüyor veya başka bir component de butonda çalışacak fonksiyonu göndereceğimizde parent dan child a o zaman da aynı ama sarmalamaya gerek yok direk gönderebiliriz provide inject olarak sadece isimlendirmeye dikkat etmek lazım provideFunc: this.provideFunc şeklinde olmayacak

Day 3

Slots

* Eğer ki component e html etiketi göndereceksek componentleri /> şeklinde değil bu şekilde yapıyoruz

<template>

  <TheModal>

       <template #title>

      <h1>Slots İle Gelen Title</h1>

    </template>

   </TheModal>

</template>

#title yerine v-slot:title olarakta kullanılabiliyor

* Title bilgisini göstereceğimiz sayfada ise

<div class="header">

    <slot name="title"></slot>

  </div>

* Buna Name Slots deniyor
* Birde Default Slot var onda ise

<template #default>

      bu slot il gösterilen default bir bilgi

    </template>

diğer componentde

  <slot/>

bir de template kullanmak yerine div de kullanabiliriz... ve slot bitti...

Yeni Yöntem Teleport

component ler arası geçişte mesela inputtaki bir değerin kaybolmaması için <keep-alive></keep-alive> ile componenti sarmalıyoruz ama kullanımı bu şekilde olacak kalmasını istediğim elementleri içerisine yazıyoruz

    <keep-alive>

      <component: is="activeComponent">

        <h3 class="bg-blue text-white">Blue</h3>

       <input/>

      </component:>

    </keep-alive>

keep-alive bize state yönetebileceğimiz iki tane lifecycle hook veriyor activated ve deactivated

<script>

export default {

       activated(){

              console.log("Blue activated");

       },

       deactivated(){

              console.log("Blue deactivated");

       }

}

</script>

Http Request

npm install -g json-server önce böyle kuruyoz sonra

json-server --watch db.json ile jsondaki içi boş olan items verisini local de görüyoruz buna api açmakta diyebiliriz

* npm i axios ile kuruyoruz
* axios.post ile items içini dolduruyoruz
* axios u her yerde import etmek yerine property gibi gösterebiliriz main.js de import axios from "axios"; sonra app.mount dan önce app.config.globalProperties.$axios = axios; yapıyoruz
* bu ayarlamayı yaptıktan sonra istediğimiz yerde this.$axios ile kullanabiliriz

**Axios Custom Instance Yapımı**

* Her yere this.$axios.get veya post localhost yazmak yerine bir instance dan çağırıyoruz src altında utils klasörü appAxios.js dosyasına

import axios from "axios";

export const appAxios = axios.create({

       baseURL: "http://localhost:3000",

       withCredentials:false,

       headers:{

              tokenX: "MyToken",

              "Content-Type": "application/json"

     }

});

import { createApp } from 'vue'

import App from './App.vue'

import router from './router';

import axios from 'axios';

import {appAxios} from './utils/appAxios';

const app = createApp(App);

app.use(router);

app.config.globalProperties.$axios = axios;

app.config.globalProperties.$appAxios = appAxios;

app.mount('#app');

Day4

Vuex

* Kendisi bir State Management
* Bütün componentlerde sayfalar arası bilgi paylaşımında kullanmalıyız
* datayı vuex demi component demi tutsam şeklinde büyük bir projeye başlarken düşün basitleştirmek için bu bilgiyi başka bir component kullanacak mı onu düşün
* npm ile kuruyoruz npm install vuex@next --save
* en başta store.js diye bir dosya oluşturuyoruz adı çokta önemli değil ama global bir store oluşturuluyor bunun içindeki bilgilerimiz bizim state olarak ifade ediliyor biz burada stateleri tanımlıyoruz
* getters ın tek işi bize state içersindeki bilgileri döndürmektir
* state içinde bir array dizisindeki her bilgiyi göstermek yerine mesela backend e sürekli password bilgisi gitsin istenmiyorsa getters içerisine fonksiyon alan bir şeydir
* ... buna javascript spred operator deniyor
* getters ı iki türlü kullanabiliriz
* her yere this.$state.getters yazmak yerine computed yazarak ve mapGetters ı import ederek ikisini birlikte kullanabiliriz ama şöyle bir durum ortaya çıkıyor sayfada computed ı kullanmak istersek mapGetters ı silmek gibi bir durumla karşılaşıyoz bunun için şöyle kullanıyoruz:

// computed: mapGetters(["woodItems"]),

  computed: {

    ...mapGetters(["woodItems","activeUser"]),

    customComputed(){

      return console.log('object :>> ');

    }

  }

* mapGetters ı array kullanım yerine object olarakta kullanabiliriz
* bir de mutations kavramı var global fonksiyon diyebiliriz database işlemi yapacaksak servise atılan işlemi beklemez async değil sync dur async işlemler için kullanılan property actions:{} aynı mutation lar gibidir ama async
* mutation lar commit action lar dispatch edilir
* veri güncelleme işi mutations ın işi olduğundan actions da güncelleme yapma context.commit("newItem",item); yap
* Tek bir store olmasını istemiyorsak o zaman modules kullanıcaz ve içerisine contact gibi js dosyaları oluşturucaz proje genelinde bir tane store oluşturulur diğerleri module olur kullanımı için oluşturduğumuz store klasöründe index.js içinde modules:{} açıyoruz
* sadece state i değil diğer modullerde de mutations falan tanımlayabiliriz

Day 5

Vue Router

* npm install vue-router@4 ile kuruyoruz
* router-link router-view ile beraber kullanılıyor ve sayfa yenilenmiyor performans katıyor router-link e active-class verip hangi sayfada olduğunu mesela button üzerinde gösterebiliriz active-class css classı
* name: "HomePage" ile route yönlendirmelerini daha kolay anlaşılır hale getiriyoruz
* Button İle Yönlendirme
* $route ile parametrelerine erişebiliriz
* Parent da methods içinde goBack diye fonksiyon yapıp içerisine this.$router.push("/about") yapıp button ile yönlendirme yapabiliriz
* name i şöyle kullanabiliriz this.$router.push({

name:"DetailPage"});

* name:"DetailPage",params:{id:e.target.value} params ile de inputtaki değeri id bilgisi olarak verebiliriz
* name params dışında query diye de parametremiz var sorgu çalıştırmak
* router işlemli projelerinde kesinlikle # işaretini kullan yoksa sayfa yenileme durumlarında 404 hatası ile karşılaşabilirsin yani createWebHistory yerine createWebHashHistory kullan ki 404 gibi sorunlarla projen karşılaşmasın

Bu Zaman Kadar ***Option Api*** den Bahsettik Şimdi Yeni Konu ***Composition Api***

Composition Api

* 1. v-model computed watcher ile izlenmesi bunlar reactivity data içinde oluşturduğumuz tüm yapılar reactive bir değişken oluyor gereksiz bir sürü değişken var reactivitye sahip olan bu da projeye yük bindirir gereksiz yükden kurtulmak için
* 2.parçala yönet tekniğini uygulamak

Vite

* Bir bundle serverdır vue-cli kendi içinde çevirme yapıyor ama şimdi tarayıcılar zaten vue-cli ın yaptığı çevirme işlemini yapıyor o yüzden vite çevirme harici işi minimum gereksinimleri sağlayalım yeter browser ın işi onda kalsın diyen bir şey sadece vue js için değil react gibi başka frameworklerde de kullanılıyor
* vite kullanmak için önce klasör açıyoruz sonra npm init vite@latest
* composition api ile option api birlikte kullanılabilir
* npm run dev ile projeyi ayağa kaldırıyoz
* setup fonksiyonu beforecreated created ve data yerine gelmiş bir fonksiyon hatta beforecreated dan daha önce çalışıyor
* setup fonksiyonu geriye bir object return etmeli çalışması için datayı diğerleri gibi replace edebilmesi için return e ihtiyacı var
* input içerisinde falan değişebilecek değerler için ref kullanıyoruz
* const title = ref("bu bir değişken başlık"); bu şekilde
* methods kullanmamıza da gerek kalmıyor setup içerisinde direk olarak fonksiyon tanımlayabiliyoruz
* setup içerisinde computed da kullanabiliriz

import {ref, computed} from "vue";

const titleLengthMessage = computed(()=>{

      return title.value.length + " adet karakter yazdınız";

    });

    return{

      titleLengthMessage,

    }

* template içerisinde gösterme aynı {{titleLengthMessage}}
* api ile yapılan işlerde watchEffect biraz hayat kurtarıcı yani bir inputta her bastığımızda api de arama yapıp apiyi patlamayı önleyebilir şöyle içerisinde onInvalidate(()=>{}) ile yani bununla kullanıcı yazmayı bitirdiğinde istek gider yazma işlemi bitti mi takip eder

 const searchText = ref("");

const isType = ref(false);

watchEffect((onInvalidate) => {

      if (searchText.value.length > 0) {

        isType.value = true;

        const typing = setTimeout(() => {

          isType.value = false;

        }, 1500);

        onInvalidate(() => clearTimeout(typing));

      }

    });

    return {

      show,

      toggleIt,

      titleLengthMessage,

      searchText,

      isType,

    };

* Setup içerisine iki tane parametre alır biri props diğeri context gelen props bilgisini kullanmak için

Script Setup

* <script setup></script> yani script artık bizim setup fonksiyonumuz olmuş oluyor bir güzel yanı da önceden return ile export ettiğimiz şeyleri şimdi otomatik kendisi export ediyor return ile içini doldurmamıza gerek kalmıyor
* Normalde script setup kullanıyorsak diğer componentte counter ı props olarak almaya gerek yok direk {{counter}} yazmak bile yeterli ama çalışmama gibi durum olursa şu kod ile de alınabilir

       const props = defineProps({counter:Number});

* props.counter olarak yazdırırız
* computed da kullanabiliriz

       const result = computed(() =>( props.counter % 2 === 0 ? "Çift":"Tek"));

* defineEmits de yapabiliriz
* ref içerisine object alması önerilmiyor o yüzden yeni konu
* Reactive
* Datanın kopyasıdır

const state = reactive({

  personal: {

    name: "Felakettin",

    surname: null,

              },

       });

       console.log(state.personal.name);

* içerisine object almak zorunda watch ile izlenecek değişkenleri yazıyoruz tek tek bir sürü değişkeni ref ile tanımlamak yerine yapıyoruz ama watch kullanacağımızda json ile bir veriyi takip ettiğimiz için watch(()=>JSON.parse(JSON.stringify(state.personal)),(newPersonal, oldPersonal) => {});
* şeklinde kullanıyoruz yoksa eski ve yeni değeri ayırt edemiyor
* https://tailwindcss.com/docs/guides/vite tailwindcss kurulumu

ASYNC Componentler

Suspense

* Kullanımı github da var isLoad ile gelen veriyi yüklenirken loading yazdırmak için kullandık ağın davranışına hızına göre div yükleniyor veriyi tamamlanmadan göstermek istemiyorsak suspense kullanıcaz is Load ile uğraşmamak için
* suspense birden fazla yazılabilir

Teleport

* başka bir componenti veya etiketleri başka bir element içerisine almak için örnek olarak

    <div id="app"></div>

    <div id="teleport-area"></div>

* template kısmında ise <teleport to="#teleport-area">div vs...</teleport>

şeklinde kullanıyoz

NameSpaced Components

* Tek bir import ile bir sürü componenti çağırmaya yarıyor mesela 20 tane componentlik bir proje olsa 20 tane componenti tek tek import etmek yerine tek import ile daha temiz bir iş yapılır

FRONT END KISMI BURAYA KADAR GERİ KALAN 8 BÖLÜM BACKEND AMA DEVAM EDEBİLMEK İÇİN ÖNCE DOCKER EĞİTİMİ İZLENMELİ