computed properties.

looping elements using v-for

```
<div id='app'>
  <ul>
    <li v-for='item in language' :key='itemid'
          {{item.name}}
    </li>
  </ul>
<script>
  const app = Vue.createApp({
    data() {
      return {
        Languages: [
          {id:1, name: 'html'},
          {id:2, name: 'css'},
          {id:3, name: 'javascript'}
        ]
```

- html
- css
- javascript.

always use key while doing lo

(or)

instedd of using seperate key co

use the index like the following e

```
<li v-for='(item, index) in language
      :key='index'>
          {{item.name}}
```

for this code there is no need of id

Components.

```
<div id:'app'>
    <appbutton> </appbutton>
</div>

<script>
  let app = {
    Components :
      'appbutton' : {
        mounted() {
            alert('hello'); }
        }
    };
  Vue.createApp(app).mount('#app');

</script>.
```

creating a custom componen named appbutton

Vue is mounted to inside d

output :

showing the alert box.

adding template

from the following code,

```
<script>
  let app = {
    components: {
      template : '<h1> hi dhinu </h1>',
      mounted() {
        alert('hello'); }
      }
    }
  };
  Vue.createApp(app).mount('#app');
</script>
```

this code work similar like before inthis code we are adding the template into our components after the alert box closing it will show the <h1> tag .

Slot mechanism.

components : {

'appbutton' : {

template : '

<button> </button> '

}

<appbutton> hello <appbutton>

output ▭

(In this code we don't put text in template

but we provide while mounding but the not recognised. For recognaising that use slot.

inside the template.

template :

' <button> <slot> </slot> </button>

now its perfect.

<appbutton> hello </appbutton>

⊗ we can also call the custom

output. hello

:)

components per file.

index.html

```
<div id = 'app'>
  <appbutton> dhin </appbutton>
</div>
<script type = 'module'>
  import appbutton from './appbutton.js'
  Vue.createApp({
    components: {
      'appbutton': appbutton
                }
  }).mount('#app');
</script>
```

appbutton.js

```
export default {
  template: <button> <slot /> </butt
  mount() {
    alert('hello');
          }
  };
```

V-model

```
<div id : 'app'>
   <input type :'text' v-model ='name' >
</div>
<script>
Vue.createApp({
    data () {
      return { name: ' ' };
    }).mount ('#app');
```

we can change the input value from the input and output using the v-model.

```
<input type : 'text' :value = 'name'
      @input = "name = $event.target.value"
```

↙

This code is also doing the same what v-model does.

Running the json file in server

```
npx json-server db.json
```

filename.

That is run in port 3000. If need to change the port,

```
npx json-server db.json -p 3001
```

port num.

<style scoped>
</style>

without scoped the style will be applyed for all the vue app.
in using scoped key the style only apply for that file only.

function operators.

```
const addCount = () => count.value ++ ;
```
→ short hand

```
function addcount() {
    count.value ++ ;
}
```
→ regular function.

```
function addcount(value) {
    return value + 1 ;
}
```
→ named function and passing variables.

Counter app

using composition api

```
<script setup>
import {ref} from 'vue';

const count = ref(0);

const addcount = () => {
    count.value ++ }

const subcount = () => {
    count.value-- }

<template>
  <main>
    <h4> count is </h4>
    <h1> {{count}} </h1>
    <button @click = 'subcount'>-
```

using option api

```
<script>
  export default ( {
    data() {
      return { count : 0 }
      },
    methods : {
      addcount () {
        this.count ++ },

      subcount () {
        this.count--}}
      }),
```

→ for the same template.

Django

python backend dev framework.

pip install django → install django.

alternate → python -m django startproject myproject

django-admin startproject myproject → cmd to create a django project.

myproject

myproject
- init - py
- asgi.py
- setting.py
- urls.py
- wsgi.py.

manage.py.

} after installed the file order look like this.

routing django apps.

python manage.py startapp myapp. → It will create a myapp folder in our proj.

inside this, myapp

> migrations
--init--.py
admin.py
app.py
models.py
tests.py
views.py

url configiration, inside this myapp folder create a file called urls.py.

in myapp/urls.py.

```
from django.urls import path
from . import views.

urlpatterns = [
    path('', views.index, name = index),
    path('download', views.downlodd, name =
                                    download),
```

the views is represent the (myapp folder)

function name contains views.py.

in myapp/views.py.

```
~~from shortcuts~~
from django.shortcuts import render
from django.http import HttpResponse

def index(request):
    return HttpResponse('<h1> hello </h1>').

def download (request):
    HttpResponse('downloadd page').
```

python manage.py runserve → to run our app.

It defaultly render django site in home but we set the httpresponse to add that

inside myproject/ urls.py ~~need~~ add this following,

```
from django.urls import path, include.

urlpatterns = [
path ('admin/', admin.site.urls),
path('', include ('myapp.urls'))
    ].
```

add this 2.

here we included the urls.py path from myapp folder.

to render html file,
add template folder that indent same as manage.py.

in order to django find template folder inside our myproject/ setting.py
in that file,

```
TEMPLATES = [
    { 'BACKEND': ......
      'DIRS': [BASE_DIR, 'templates'],
```

add this line.

to after creating some html file in the templates folder,
in our views.py change the index func like following,

```
def index(request):
    ~~render~~
    return render (request, 'index.html').
```

to pass dynamic data.

```
name = 'dhin'
return render(request, 'index.html', {'name': name})
```

we can get that in {{ name }} in html jinja.

To add external static css file.

create a new folder static.

in setting.py.

import os

below STATIC URL

```
STATICFILES_DIRS = (os.path.join(BASE_DIR, 'static'),)
```

→ add this 2.

in html page,

```
{% load static %}
<link rel='stylesheet' href="{% static 'styles.css' %}">
```

add this line in html file to work.

Django models.

Configure Database.

in our app folder we have the model.py file.

in models.py

```
from django.db import models

class Feature:
    id : int
    name : str
    detail : str
    is_true : bool
```

in our view.py file, we can import the class as use that,

```
from .models import Feature

def index(request):
    Feature()
```

we can reuse the class, and pass the data to our temp

Admin panel and manipulation of Database.

if default when we create django proj there in in sqlite db avail

in project/setting.py.

INSTALLED-APPS = [... , 'myapp'] → need to add our app project like this before intergate db in models.

in models.py

```
class Features (models.Model) :
    name = models.CharField (max_length = 100)
    details = models.CharField (max_length = 500).
```

python manage.py makemigrations ← it will setup the files for migration from our modelfile.

python manage.py migrate ← it will migrate all the files to our sqlite db.

if you view the db, you can see the name, details column

admin control.

type 127.0.0.1:8000/admin,

it will ask username and pass
to get credentails,

python manage.py createsuperuser → it will ask mail and pass,

after login you can see admin panel.

before we migrate few things but that not shown in the admin panel,
to make visible
mypapp | admin.py.

```
from .models import Features

admin.site.register(Features).
```

in our views.py.

```
from .models import Features

def index(request):
    name = 'abhin'
    features = Features.objects.all()
```

we can get the data like this,

user registeration in Django.

1. price update tool.
2. RIM price sent.
3. SEO Tool.
4. Digital Marketing Agency info scraper.
5. Email Scraper from website.
6. Vector db builder
7. Facebren.