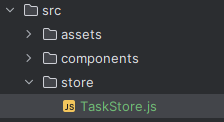
Step 1: npm install pinia

Step 2: Go to **main.js**, import korte hobe **createPinia** ke and then just **use** korte hobe, full main.js snippet,

import {createApp} from 'vue'  
import {createPinia} from "pinia";  
  
import *App* from './App.vue'  
  
import '@/assets/main.css';  
  
const app = createApp(*App*);  
  
app.use(createPinia());  
  
app.mount('#app');

Step 3: **src** er under e ekta **stores** naam e ekta folder create kora holo, and **stores** folder er under e **TaskStore.js** naam e ekta folder create kora holo,



Step 4: Now from **TaskStore.js,**

import { defineStore } from "pinia";  
  
export const *useTaskStore* = defineStore('taskStore', {  
 state: () =>({  
   
 })  
})

* defineStore diye state management ke address kore and **‘taskStore’** eta diye naam ta ke define kore current store er

Example with values,

import { defineStore } from "pinia";  
  
export const *useTaskStore* = defineStore('taskStore', {  
 state: () =>({  
  
 tasks: [  
 {id : 1, title: 'Task 1', isFav: false},  
 {id : 2, title: 'Task 2', isFav: true},  
 ],  
 name: "Yoshi",  
  
 })  
})

Step 5: Accessing Store State (Composition API)

<script>  
  
import {*useTaskStore*} from "@/store/TaskStore";  
  
export default {  
 name: 'App',  
  
 setup() {  
 const taskStore = *useTaskStore*();  
 return {  
 taskStore,  
 }  
 },  
   
 components: {}  
}  
</script>

Accessing Store State (Options API)

<script>  
  
import {*useTaskStore*} from "@/store/TaskStore";  
  
export default {  
 name: 'App',  
  
  
 data(){  
 return {  
 taskStore: *useTaskStore*()  
 }  
 },  
  
 components: {}  
}  
</script>

Step 6: Accessing value from for example **App.vue**,

<div>  
 <p>{{ taskStore.tasks }}</p>  
</div>

**#Getters**

Getters er function lekhar style as same as **vuex**. Code snippet from TaskStore.js,

getters: {  
 favs(){  
 return this.tasks.filter(task => task.isFav)  
 }  
}

Full Snippet,

import { defineStore } from "pinia";  
  
export const *useTaskStore* = defineStore('taskStore', {  
 state: () =>({  
  
 tasks: [  
 {id : 1, title: 'Task 1', isFav: false},  
 {id : 2, title: 'Task 2', isFav: true},  
 ]  
  
 }),  
 getters: {  
 favs(){  
 return this.tasks.filter(task => task.isFav)  
 }  
 }  
})

Now from **App.vue**, etake **vuex** er moto this.$getters evabe pull kora lagbe na, normally variable er moto evabe likhlei fetch korte parbe,

<p>{{ taskStore.favs }}</p>

* **'taskStore'** hocche store tar naam and **‘favs’** hocche getters er moddher function.

**#Actions**

Actions er kaj ta mutations er motoi, but pinia te kono mutations nei.

Now from **TaskStore.js**, the action block,

actions: {  
 setFav(value) {  
 let task = this.tasks.find(task => task.id === value);  
 task.isFav = !task.isFav;  
 },  
 removeTask(value) {  
 this.tasks = this.tasks.filter(task => task.id !== value);  
 }  
}

And here is a snippet from **TaskDetails.vue**,

<i class="material-icons" @click="taskStore.removeTask(task.id)">delete</i>  
<i class="material-icons" @click="taskStore.setFav(task.id)" :style="{color: task.isFav ? 'red' : ''}">favorite</i>

Full code snippet,

<template>  
  
 <div class="task">  
 <h3>{{ task.title }}</h3>  
 <div class="icons">  
 <i class="material-icons" @click="taskStore.removeTask(task.id)">delete</i>  
 <i class="material-icons" @click="taskStore.setFav(task.id)" :style="{color: task.isFav ? 'red' : ''}">favorite</i>  
 </div>  
 </div>  
  
</template>  
  
<script>  
  
 import {*useTaskStore*} from "@/store/TaskStore";  
  
 export default {  
 props: ['task'],  
  
 data() {  
 return {  
 taskStore: *useTaskStore*(),  
 }  
 }  
}  
</script>