



Vue js

1. Apa itu vue js

Vue.js adalah sebuah framework JavaScript yang digunakan untuk membangun antarmuka pengguna (UI) pada aplikasi web. Ini adalah salah satu framework JavaScript yang paling populer untuk pengembangan aplikasi front-end.

Dengan menggunakan Vue.js, pengembang dapat membuat aplikasi web interaktif dengan mudah. Framework ini memungkinkan pengguna untuk mengelola komponen-komponen UI secara terstruktur, membuat pengembangan aplikasi web lebih terorganisir dan mudah dipelihara.

2. Apa kelebihan dari vue js?

1. Ringan dan Mudah Dipelajari: Vue.js memiliki ukuran file yang kecil, sehingga memungkinkan aplikasi web untuk memuat lebih cepat. Selain itu, sintaks Vue.js yang sederhana dan mudah dipahami membuatnya cocok untuk pemula dan pengembang yang baru memulai.
2. Reaktivitas: Vue.js menggunakan reaktivitas dalam sistemnya. Ini berarti ketika data berubah, tampilan akan secara otomatis diperbarui. Ini membuat pengelolaan state dan manipulasi tampilan menjadi lebih mudah.
3. Berbasis component: Vue.js mempromosikan penggunaan komponen dalam pengembangan. Komponen-komponen ini dapat digunakan kembali dan diintegrasikan dengan mudah, membuat pengembangan lebih cepat dan mudah dipelihara.
4. Dokumentasi yang Baik: Vue.js memiliki dokumentasi yang sangat baik dan lengkap. Dokumentasi yang baik adalah sumber daya yang penting bagi pengembang, terutama bagi pemula, karena membantu memahami konsep-konsep dasar dengan jelas.

3. Tools

1. Vs code
2. Node js

Dasar vue

1.npm

merupakan pengelola package untuk JavaScript yang dapat memudahkan kita dalam mengelola package yang tersedia pada [npmjs.com](https://www.npmjs.com). NPM ini merupakan standard package manager yang disediakan oleh Node.js dan sudah otomatis terpasang ketika memasang Node.js.

Perintah di npm :

1.npm init

Membuat file *package.json* pada project

2.npm install <package-name>

di gunakan untuk install package,bisa juga menggunakan alis seperti : **npm i /npm add**

3.npm run

untuk menjalankan perintah dalam objek script yang ada di package.json

4.npm uninstall <package-name>

untuk menghapus package yang di install

2.Install vue js

Jalan perintah

```
npm create vue@latest
```

setelah install selesai jalankan

```
npm install dan npm run dev
```

3.Creating a application

CreateApp()

Aplikasi vue akan berjalan ketika function **createApp()** dijalankan.kita perlu mem parsing component ke dalam function **createApp()** dengan mengimport component lalu memasukan nya ke dalam function **createApp()**.

Mount()

Aplikasi vue tidak akan bisa di render ketika function **mount()** belum di panggil.aplikasi vue perlu merender root component ke dalam sebuah element dengan memasukan selector ke dalam function **mount()**.

contoh :

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

4.Templating

Sintaks template di vue js di gunakan untuk mengikat/menampilkan data dalam komponen ke html sehingga perubahan yang terjadi di data dalam komponen akan otomatis mengubah tampilan di html.

Templating dengan Mustache

```
<template>
<p>{{nama}}</p>
</template>

<script>
export default{
  data(){
    return {
      nama: 'WUDI'
    }
  }
}
</script>
```

Raw html

Jika menggunakan Mustache data di komponen akan menjadikan data tersebut sebagai string.jika kamu ingin menjadikan data nya sebagai html gunakan **v-html**.

```

<template>
  <div v-html="nama"></div>
</template>
<script >
export default{
  data(){
    return {
      nama:'<h1>Wudi</h1>'
    }
  }
}
</script>

```

5.Attribute data Binding

di gunakan untuk memberikan value ke attribute html menggunakan property yang ada di dalam komponent.

```

<style>
  .primary{
    color:blue;
  }
</style>
<script >
export default{
  data(){
    return {
      objectProperty:{
        id:'1',
        class:'primary'
      },
      nonaktifkan:true
    }
  }
}
</script>
<template>
  <button :disabled="nonaktifkan">submit</button>
  <h1 v-bind="objectProperty">Hallo</h1>
</template>

```

6.JavaScript Expressions

Di vue javascript expression bisa di gunakan di dalam mustache dan dalam data binding.

```
<template>
  {{number+1}}
  {{ isTrue ? 'YES' : 'NO' }}
  {{ Date.now() }}
  <div :id="`list-${number}`"></div>
</template>
<script>
export default{
  data(){
    return {
      number:0,
      isTrue:true,
      message:'thor'
    }
  }
}
</script>
```

Directive

adalah sebuah attribute spesial di vue yang di awal dengan v- contoh seperti **v-bind**,**v-html** dan **v-if** dan lainnya.directive vue akan selalu menerima data dari properti component yang data tersebut akan secara langsung berubah jika data properti di component berubah.

7.Directive v-if

Disini directive v-if akan menghapus atau menambahkan element,tergantung kondisi dari data properti di component.

```
<template>
<p v-if="show">Element di tampilkan</p>
</template>
<script>
export default{
  data(){
    return {
      show:true
    }
  }
}
</script>
```

8.Computed dan method

adalah sebuah objek yang di gunakan untuk membuat function dan logic” di di vue js.perbedaan antara computed dan method ialah :

function di dalam computed tidak akan dijalankan jika nilai di dalam function computed telah mengalami perubahan.

function di dalam method akan terus di jalankan walaupun nilai dalam function tersebut telah berubah.

```

<template>
<button @click="countNumberMethod">{{countMethod}}</button>
  <button @click="countNumber">{{count}}</button>
</template>
<script>
export default{
  data(){
    return {
      countMethod:0,
      count:0,
    }
  },
  computed:{
    countNumber(){
      return this.count++;
    }
  },
  methods:{
    countNumberMethod(){
      return this.countMethod++;
    }
  }
}
</script>

```

9. Class dan style binding

Kita bisa mem parsing seubah object ke class dengan data binding.

script di bawah akan menambahkan seubah nama class ke attribut class ketika data dari properti yang di berikan bernilai true dan jika element yang di binding sudah mempunyai class sebelumnya maka class di element akan di gabungkan dengan class attribut yang di binding.

```

<style>
  .active{
    color:blue;
  }
  .fs{
    font-size:30px;
  }
  .text-transform{
    text-transform: uppercase;
  }
</style>
<template>
<ul>
<li class="fs" :class="{active:isActive,'text-transform':isUppercase}">test</li>
</ul>
</template>
<script>
export default{
  data(){
    return {
      isActive:true,
      isUppercase:true
    }
  },
}
</script>

```

```

<template>
<h1 :style="{color:textRed,fontSize}">Thor</h1>
</template>
<script>
export default{
  data(){
    return {
      textRed:'red',
      fontSize:'50px'
    }
  }
}
</script>

```

10. Conditional rendering

terdapat 2 type untuk perkondisian di vue yaitu menggunakan v-if dan v-show.

perbedaan dari kedua nya ialah :

v-if

jika bernilai true maka element akan di render dan jika false element tidak akan di render.

v-show

element akan selalu di render ke dalam html. jika bernilai false element html hanya akan di hidden saja menggunakan **display : none**


```

<template>
  <div v-if="nilai >= 90">
    A
  </div>
  <div v-else-if="nilai <= 90 && nilai >=70">
    B
  </div>
  <div v-else-if="nilai >= 70 && nilai <=80">
    C
  </div>
  <div v-else>
    BAD
  </div>
</template>
<script>
export default{
  data(){
    return {
      nilai:100
    }
  }
}
</script>

```

```

<template>
  <div v-show="nilai <= 90">
    <p>A</p>
  </div>

</template>
<script>
export default{
  data(){
    return {
      nilai:100
    }
  }
}
</script>

```

11.List rendering

List rendering di gunakan untuk looping array atau object dari data property component di vue ke html.

```

<template>
<ul>
  <li v-for="(item,index) in class" :key="index">{{item}}</li>
  <li v-for="(item,index) in objectClass" :key="index">{{item.class}}</li>
  <li v-for="(i,index) in 10" :key="index">{{i}}</li>
  <li v-for="(item,index) in class" :key="index">
    <p v-if="item !='XB'">{{item}}</p>
  </li>
</ul>
</template>

<script>
export default{
  data(){
    return {
      class:['XA','XB','XC'],
      objectClass:[{
        class:'XA'
      },{
        class:'XB'
      }]
    }
  }
}
</script>

```

12.Event handling

Digunakan untuk menjalankan event di javascript seperti menjalankan event click,prevent.

untuk menggunakan event bisa menggunakan directive **v-on:click="handler"** atau bisa menggunakan **@click="handler"**.

handler pada directive bisa di isi method atau data property komponent.jika directive menggunakan methods di sebut sebagai **method handler** atau jika directive menggunakan data property di komponent di sebut sebagai **inline handler**.

form binding

untuk melakukan binding menggunakan element input,select,textarea bisa menggunakan v-model.

v-model adalah syntax untuk memperbarui data property setiap kali ada perubahan data yang di lakukan oleh user.

```

<template>
  <!-- input -->
  <input v-model="nama">
    <textarea v-model="nama" placeholder="Nama"></textarea>
    {{nama}}
  <!-- checked -->
  <input type="checkbox" id="checkbox" v-model="checked">
  <label for="checkbox">{{ checked }}</label>

  <!-- radio -->
  <br>
  <input type="radio" id="one" value="weekend" v-model="isWeekend">
  <label for="one">weekend</label>
  <input type="radio" id="two" value="kerja kerja" v-model="isWeekend">
  <label for="two">Kerja kerja</label>
  <span>Apakah libur? {{ isWeekend }}</span>

  <!-- selected -->
  <select v-model="select">
    <option disabled value="">Pilihan kamu</option>
    <option>Aku</option>
    <option>Dia</option>
  </select>
  <span>Pilih siapa? {{ select }}</span>
</template>

<script>
export default{
  data(){
    return {
      nama:'',
      checked:false,
      isWeekend:'',
      select:''
    }
  }
}
</script>

```

```

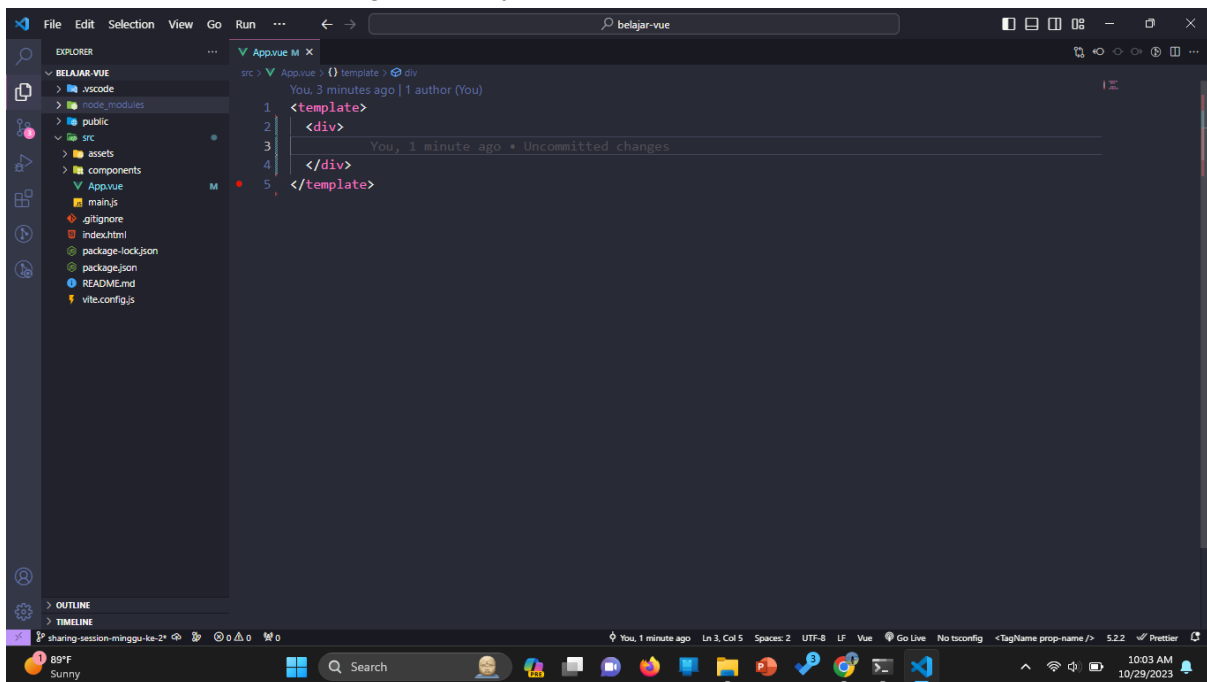
<template>
  <!-- inline handler -->
  <button @click="count++">count {{count}}</button>
  <!-- methods handler -->
  <button @click="eventCount()">count {{count2}}</button>
  <!-- mengirim params ke method -->
  <button @click="eventCountParams(3)">count {{count3}}</button>
</template>
<script>
export default{
  data(){
    return{
      count:0,
      count2:0,
      count3:0
    }
  },
  methods:{
    eventCount(){
      this.count2++
    },
    eventCountParams(count){
      this.count3+=count;
    }
  }
}
</script>

```

13. latihan todolist

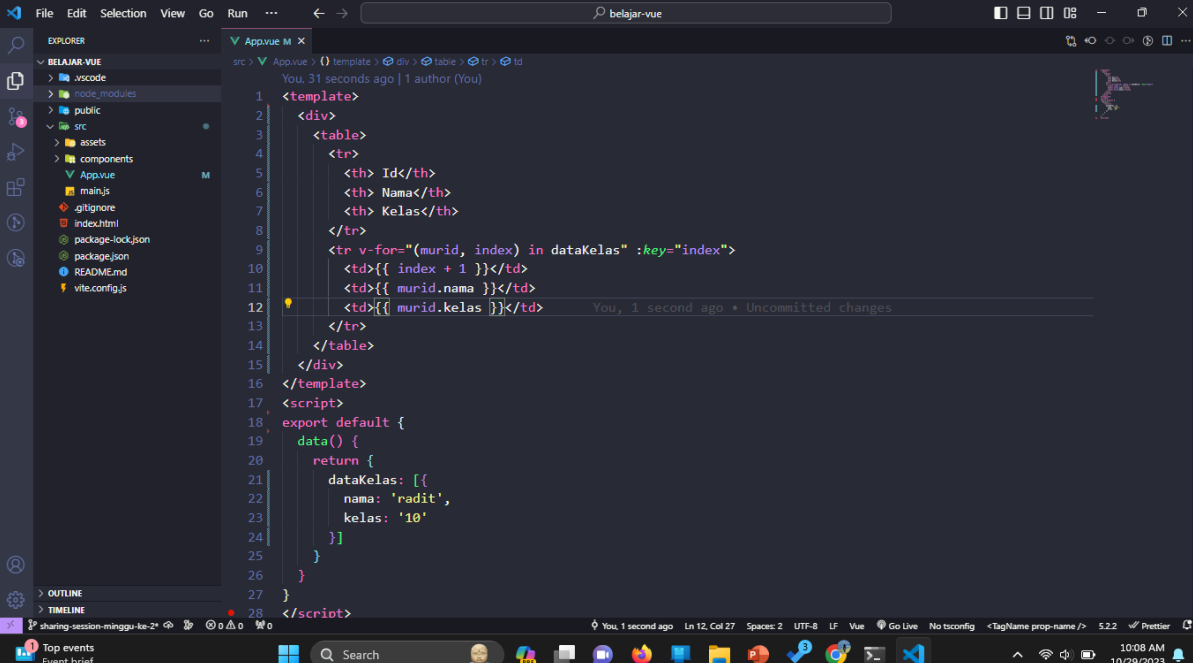
Buat project baru menggunakan vue js. lalu ikuti tutorial di bawah ini

buka file App.vue lalu kosongkan file nya



-membuat daftar siswa

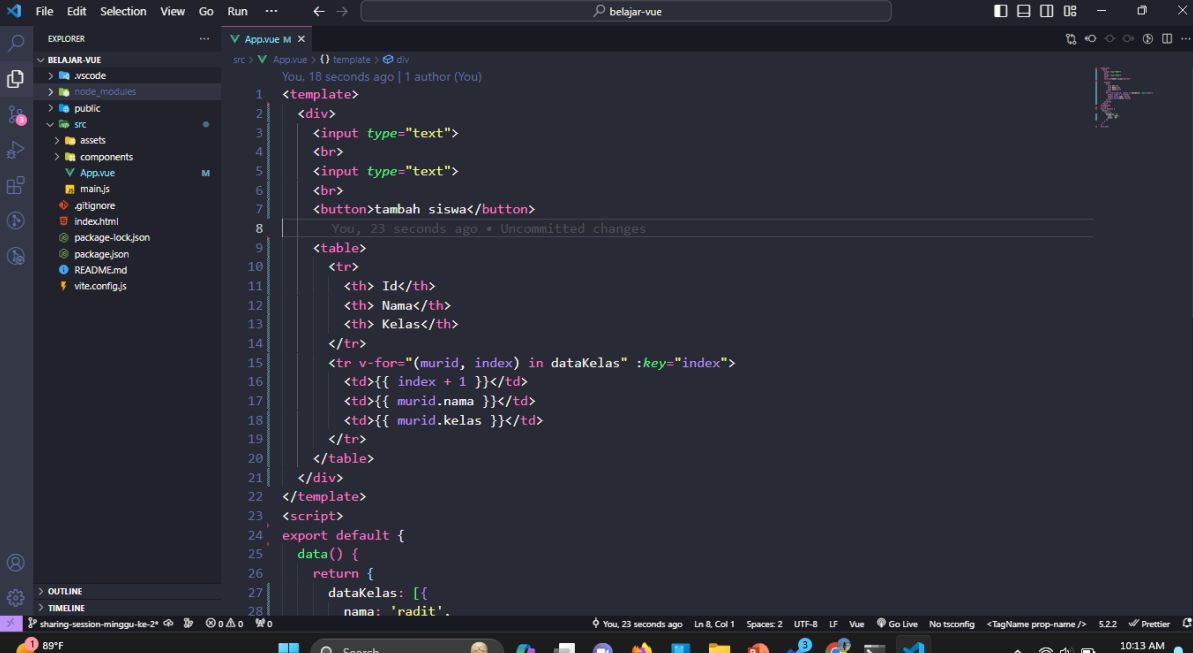
masukan script berikut di file App.vue



```
1 <template>
2   <div>
3     <table>
4       <tr>
5         <th> Id</th>
6         <th> Nama</th>
7         <th> Kelas</th>
8       </tr>
9       <tr v-for="(murid, index) in dataKelas" :key="index">
10        <td>{{ index + 1 }}</td>
11        <td>{{ murid.nama }}</td>
12        <td>{{ murid.kelas }}</td>
13      </tr>
14    </table>
15  </div>
16 </template>
17 <script>
18 export default {
19   data() {
20     return {
21       dataKelas: [
22         {
23           nama: 'radit',
24           kelas: '10'
25         }
26       ]
27     }
28   }
29 </script>
```

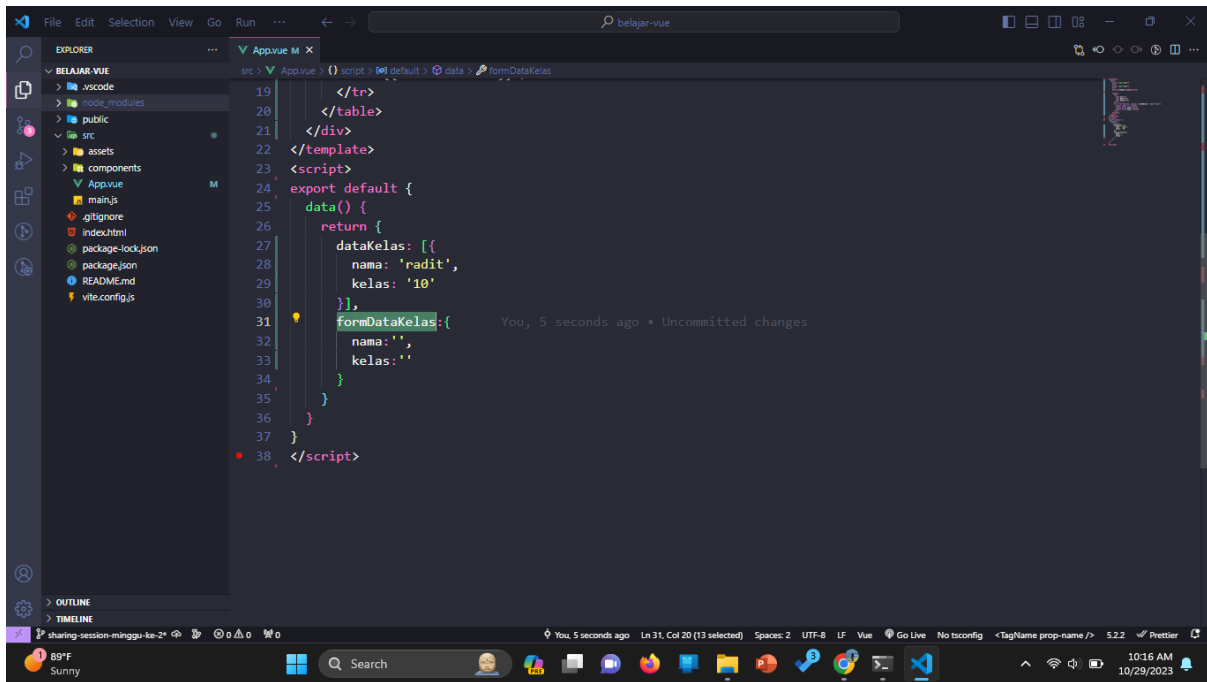
-membuat tambah siswa

buat form sederhana untuk tambah siswa di atas element table



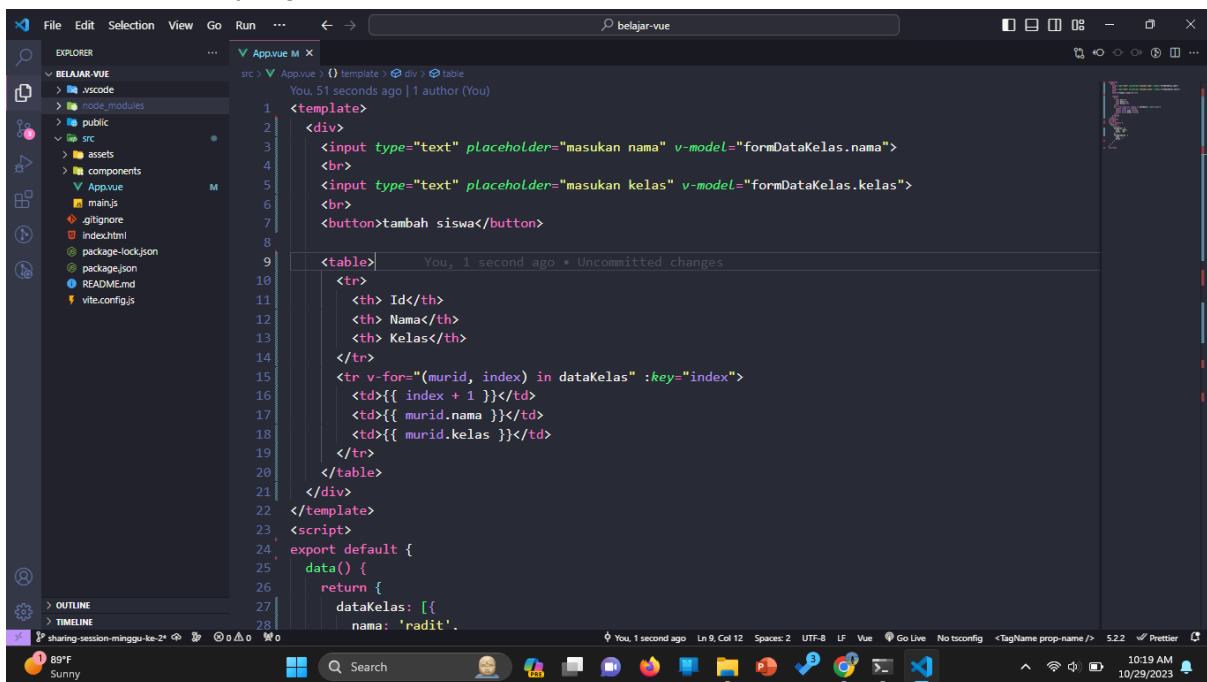
```
1 <template>
2   <div>
3     <input type="text">
4     <br>
5     <input type="text">
6     <br>
7     <button>tambah siswa</button>
8   <table>
9     <tr>
10      <th> Id</th>
11      <th> Nama</th>
12      <th> Kelas</th>
13    </tr>
14    <tr v-for="(murid, index) in dataKelas" :key="index">
15      <td>{{ index + 1 }}</td>
16      <td>{{ murid.nama }}</td>
17      <td>{{ murid.kelas }}</td>
18    </tr>
19  </table>
20 </div>
21 </template>
22 <script>
23 export default {
24   data() {
25     return {
26       dataKelas: [
27         {
28           nama: 'radit',
29           kelas: '10'
30         }
31       ]
32     }
33   }
34 </script>
```

lalu buat data object form dataKelas yang di dalamnya mempunyai property nama, kelas untuk menampung data yang akan di input ke dataKelas.



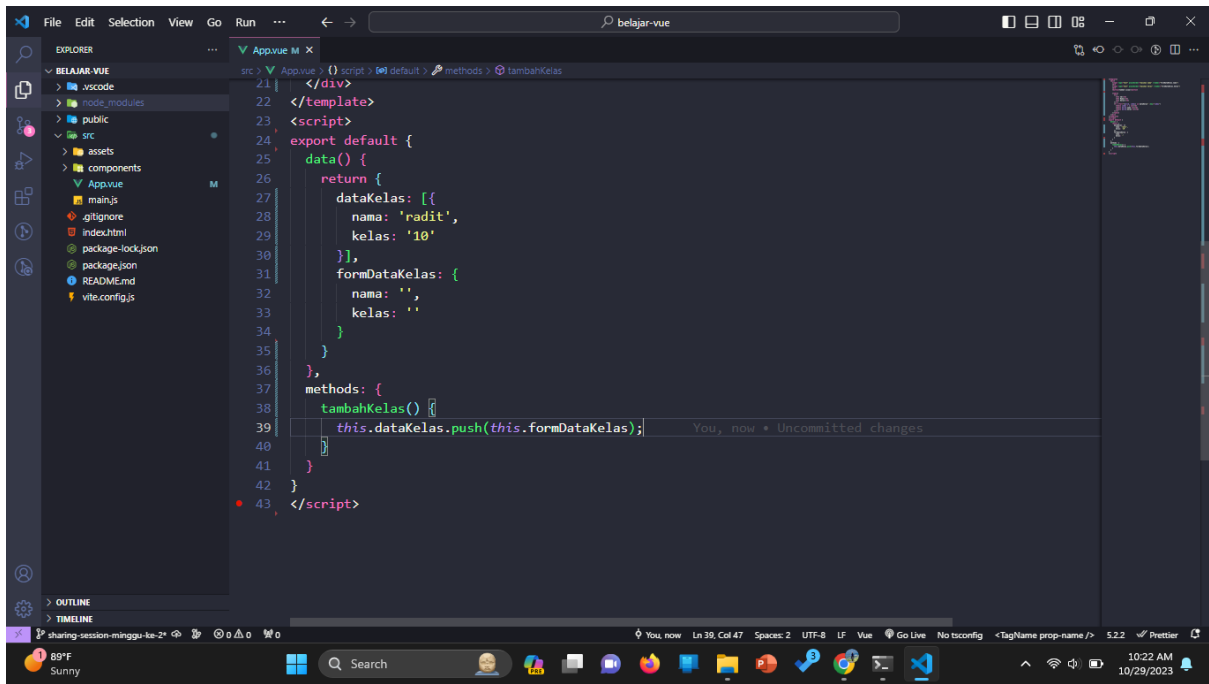
```
19 </tr>
20 </table>
21 </div>
22 </template>
23 <script>
24 export default {
25   data() {
26     return {
27       dataKelas: [{
28         nama: 'radit',
29         kelas: '10'
30       }],
31       formDataKelas: {
32         nama: '',
33         kelas: ''
34       }
35     }
36   }
37 }
38 </script>
```

setelah membuat object formDataKelas buat v-model untuk setiap inputan agar bisa memasukkan data yang di input ke dalam formDataKelas



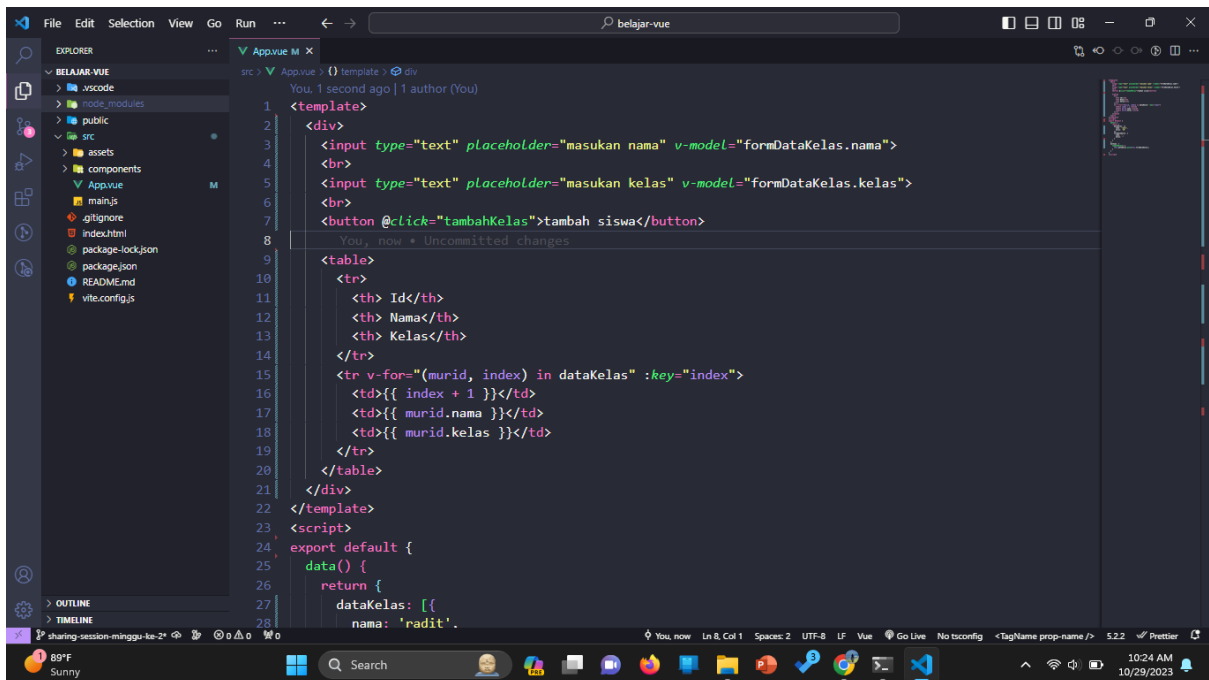
```
1 <template>
2 <div>
3   <input type="text" placeholder="masukan nama" v-model="formDataKelas.nama">
4   <br>
5   <input type="text" placeholder="masukan kelas" v-model="formDataKelas.kelas">
6   <br>
7   <button>tambah siswa</button>
8 </div>
9 <table>
10 <tr>
11   <th>Id</th>
12   <th>Nama</th>
13   <th>Kelas</th>
14 </tr>
15 <tr v-for="(murid, index) in dataKelas" :key="index">
16   <td>{{ index + 1 }}</td>
17   <td>{{ murid.nama }}</td>
18   <td>{{ murid.kelas }}</td>
19 </tr>
20 </table>
21 </div>
22 </template>
23 <script>
24 export default {
25   data() {
26     return {
27       dataKelas: [{
28         nama: 'radit',
29         kelas: '10'
30       }],
31       formDataKelas: {
32         nama: '',
33         kelas: ''
34       }
35     }
36   }
37 }
38 </script>
```

lanjut setelah membuat v-model, buat sebuah methods untuk melakukan aksi menambahkan data, untuk menambahkan data yang akan di input ke dalam dataKelas



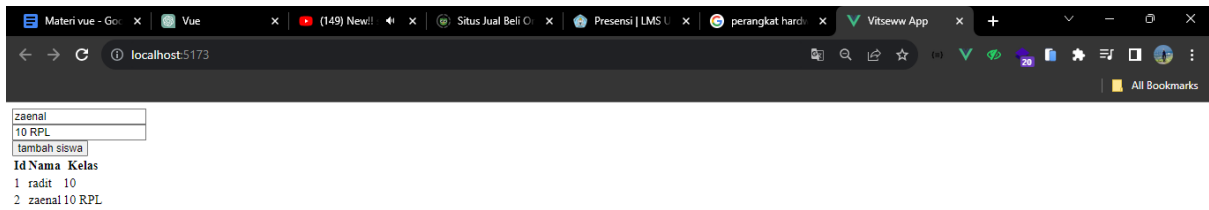
```
21 </div>
22 </template>
23 <script>
24 export default {
25   data() {
26     return {
27       dataKelas: [{
28         nama: 'radit',
29         kelas: '10'
30       }],
31       formDataKelas: {
32         nama: '',
33         kelas: ''
34       }
35     },
36   },
37   methods: {
38     tambahKelas() {
39       this.dataKelas.push(this.formDataKelas);
40     }
41   }
42 }
43 </script>
```

setelah methods nya di buat,selanjutnya beri button tambah siswa sebuah event click,yang jika button siswa di click maka akan menjalankan methods tambahKelas.



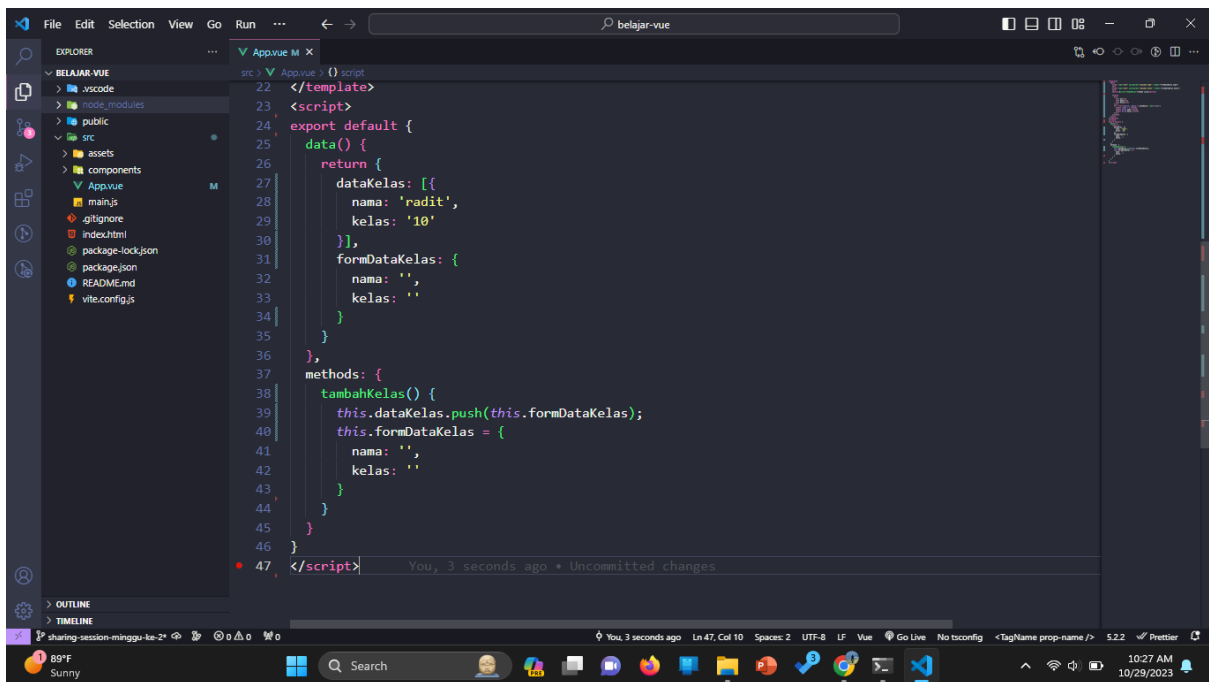
```
1 <template>
2 <div>
3   <input type="text" placeholder="masukan nama" v-model="formDataKelas.nama">
4   <br>
5   <input type="text" placeholder="masukan kelas" v-model="formDataKelas.kelas">
6   <br>
7   <button @click="tambahKelas">tambah siswa</button>
8
9   <table>
10    <tr>
11      <th>Id</th>
12      <th>Nama</th>
13      <th>Kelas</th>
14    </tr>
15    <tr v-for="(murid, index) in dataKelas" :key="index">
16      <td>{{ index + 1 }}</td>
17      <td>{{ murid.nama }}</td>
18      <td>{{ murid.kelas }}</td>
19    </tr>
20  </table>
21 </div>
22 </template>
23 <script>
24 export default {
25   data() {
26     return {
27       dataKelas: [{
28         nama: 'radit',
29         kelas: '10'
30       }],
31       formDataKelas: {
32         nama: '',
33         kelas: ''
34       }
35     },
36   },
37   methods: {
38     tambahKelas() {
39       this.dataKelas.push(this.formDataKelas);
40     }
41   }
42 }
43 </script>
```

jika kita coba maka form tambah siswa berhasil menambahkan data siswa baru



namun, data nya masih ada ketika kita meng input data. bagaimana agar data nya kosong kembali?

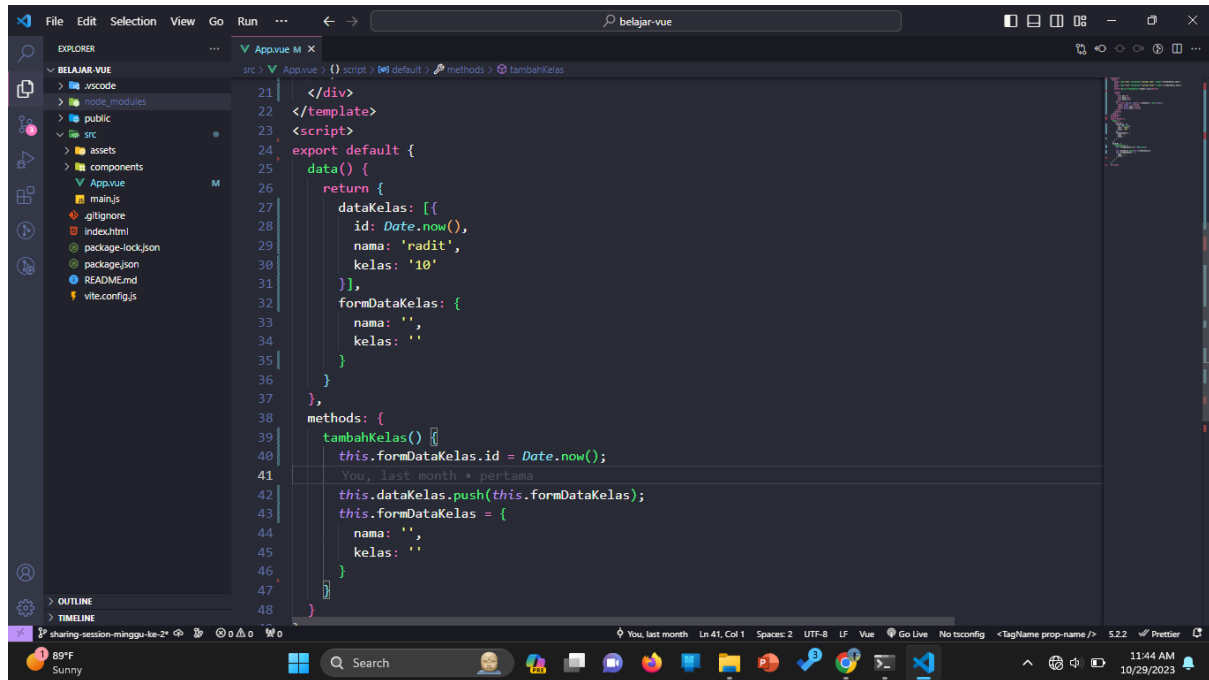
caranya, buat kembali object formDataKelas menjadi kosong, seperti ini



-membuat hapus data siswa

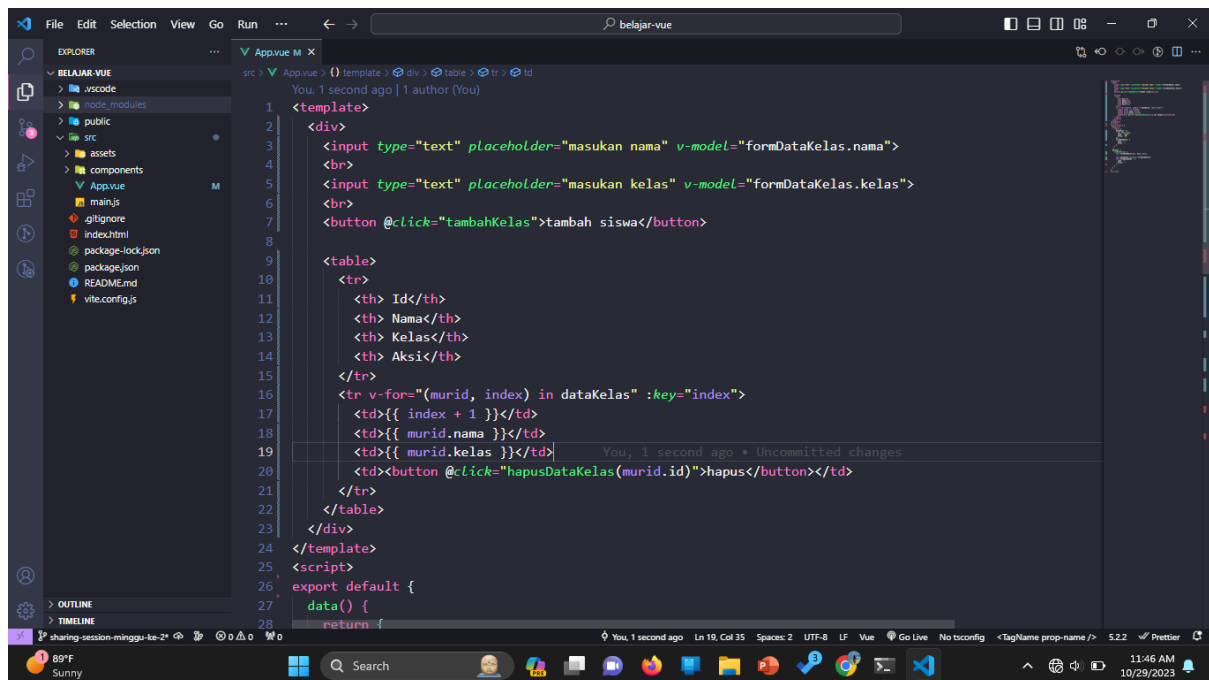
untuk membuat hapus data siswa pertama kita ubah dahulu dataKelas dengan menambahkan object id yang di isi dengan timestamp agar setiap data mempunyai id yang unik.

tambahkan object id kedalam dataKelas dan tambah object id ke formDataKelas ketika menambahkan data di methods tambahKelas



```
21 </div>
22 </template>
23 <script>
24 export default {
25   data() {
26     return {
27       dataKelas: [{
28         id: Date.now(),
29         nama: 'radit',
30         kelas: '10'
31       }],
32       formDataKelas: {
33         nama: '',
34         kelas: ''
35       }
36     }
37   },
38   methods: {
39     tambahKelas() {
40       this.formDataKelas.id = Date.now();
41       You, last month + pertama
42       this.dataKelas.push(this.formDataKelas);
43       this.formDataKelas = {
44         nama: '',
45         kelas: ''
46       }
47     }
48   }
49 }
```

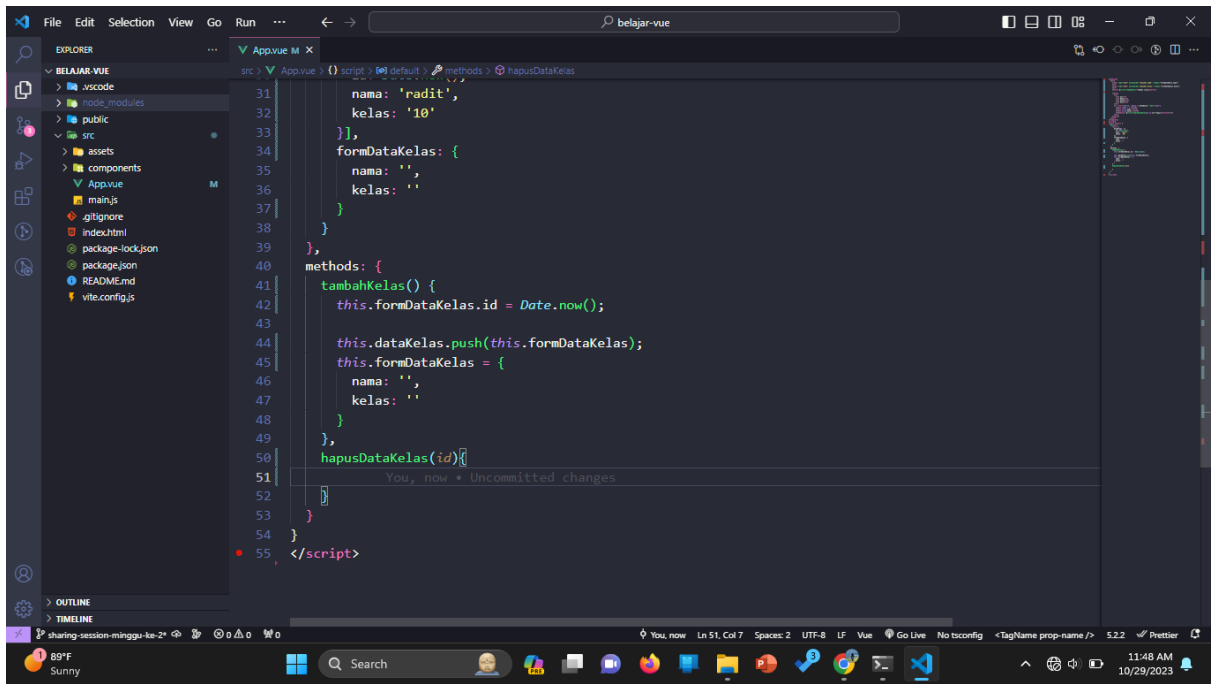
setelah itu buat button di dalam table untuk menghapus data



```
1 <template>
2   <div>
3     <input type="text" placeholder="masukan nama" v-model="formDataKelas.nama">
4     <br>
5     <input type="text" placeholder="masukan kelas" v-model="formDataKelas.kelas">
6     <br>
7     <button @click="tambahKelas">tambah siswa</button>
8   </div>
9   <table>
10    <tr>
11      <th>Id</th>
12      <th>Nama</th>
13      <th>Kelas</th>
14      <th>Aksi</th>
15    </tr>
16    <tr v-for="(murid, index) in dataKelas" :key="index">
17      <td>{{ index + 1 }}</td>
18      <td>{{ murid.nama }}</td>
19      <td>{{ murid.kelas }}</td>
20      <td><button @click="hapusDataKelas(murid.id)">hapus</button></td>
21    </tr>
22  </table>
23 </div>
24 </template>
25 <script>
26 export default {
27   data() {
28     return {
29       dataKelas: [{
30         id: Date.now(),
31         nama: 'radit',
32         kelas: '10'
33       }],
34       formDataKelas: {
35         nama: '',
36         kelas: ''
37       }
38     }
39   },
40   methods: {
41     tambahKelas() {
42       this.formDataKelas.id = Date.now();
43       this.dataKelas.push(this.formDataKelas);
44       this.formDataKelas = {
45         nama: '',
46         kelas: ''
47       }
48     },
49     hapusDataKelas(id) {
50       // Implementasi metode hapusDataKelas
51     }
52   }
53 }
```

buat juga event click untuk menjalankan methods hapusDataKelas yang mempunyai params id,dimana dengan id tersebut kita dapat tahu data mana yang hapus di hapus.

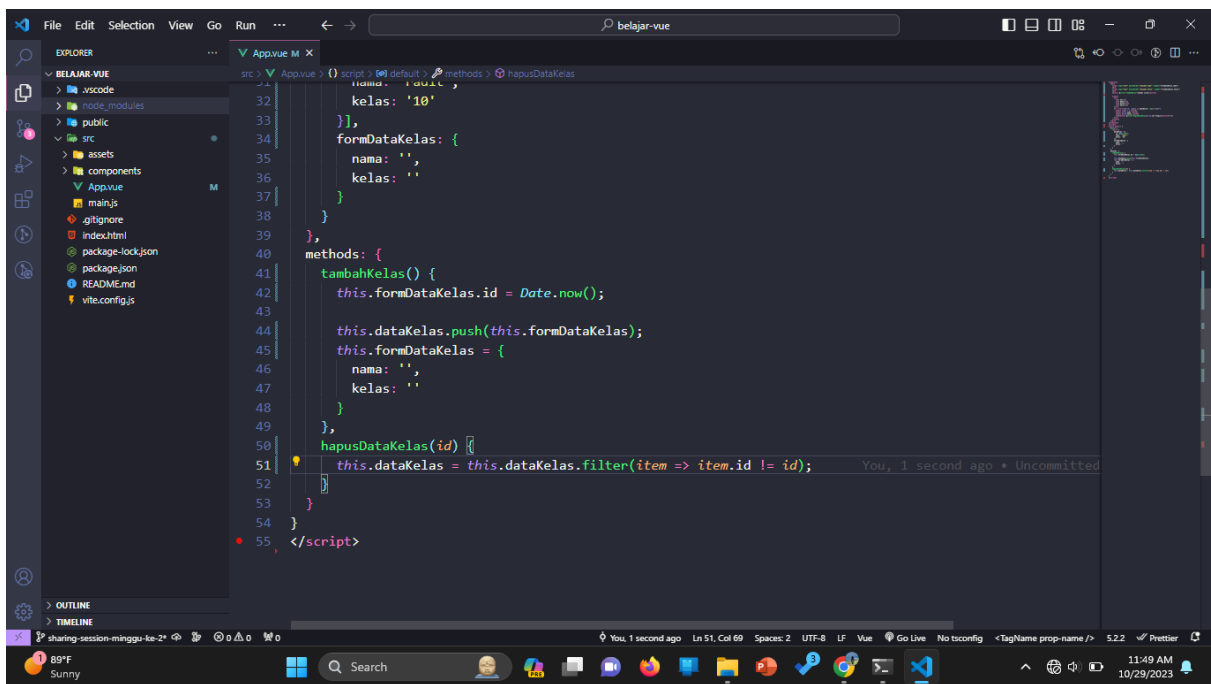
selanjutnya buat methods hapusDataKelas



The screenshot shows the VS Code editor with the file explorer on the left displaying the project structure. The main editor window shows the `App.vue` file. The `data` property is defined with `nama: 'radit'` and `kelas: '10'`. The `formDataKelas` object is also defined with `nama: ''` and `kelas: ''`. The `methods` object contains two functions: `tambahKelas()` and `hapusDataKelas(id)`. The `tambahKelas()` function generates a new ID using `Date.now()`, pushes the new data to `this.dataKelas`, and resets the `formDataKelas` object. The `hapusDataKelas(id)` function is currently empty, with a comment indicating it needs to be implemented.

```
31 |   nama: 'radit',
32 |   kelas: '10'
33 | },
34 |   formDataKelas: {
35 |     nama: '',
36 |     kelas: ''
37 |   }
38 | },
39 |
40 |   methods: {
41 |     tambahKelas() {
42 |       this.formDataKelas.id = Date.now();
43 |
44 |       this.dataKelas.push(this.formDataKelas);
45 |       this.formDataKelas = {
46 |         nama: '',
47 |         kelas: ''
48 |       }
49 |     },
50 |     hapusDataKelas(id){
51 |       // You, now * Uncommitted changes
52 |     }
53 |   }
54 | }
55 | </script>
```

lalu buat script untuk untuk menghapus data di array DataKelas sesuai dengan id nya.



The screenshot shows the same VS Code editor with the `App.vue` file. The `hapusDataKelas(id)` function has been updated to use the `filter` method to remove the data with the specified ID from the `dataKelas` array. The status bar at the bottom indicates the file has been modified 1 second ago.

```
32 |     kelas: '10'
33 |   },
34 |   formDataKelas: {
35 |     nama: '',
36 |     kelas: ''
37 |   }
38 | },
39 |
40 |   methods: {
41 |     tambahKelas() {
42 |       this.formDataKelas.id = Date.now();
43 |
44 |       this.dataKelas.push(this.formDataKelas);
45 |       this.formDataKelas = {
46 |         nama: '',
47 |         kelas: ''
48 |       }
49 |     },
50 |     hapusDataKelas(id) {
51 |       this.dataKelas = this.dataKelas.filter(item => item.id !== id);
52 |     }
53 |   }
54 | }
55 | </script>
```

lihat browser lalu click button hapus,datapun akan terhapus.