
Vue.js

Menambahkan Library (Package)

Kita bisa menambahkan package atau library baru kedalam vue. Untuk menambahkan library, pacakage atau dependency kedalam project vue. Kita akan menggunakan yang namanya NPM

Sebagai Contoh kita akan coba menambahkan Bootstrap ke dalam Vue Js menggunakan NPM (Node Package Manager). Bisa menggunakan cmd dari window, terminal dari mac atau terminal dari textEditor yang kita gunakan

Untuk menginstal bootstrap cukup menjalankan perintah yaitu

```
npm install --save bootstrap
npm install --save @popperjs.core
```

Kita perlu menambahakn atau import kedua installan tersebut agar bisa digunakan secara global

```
import "bootstrap/dist/css/bootstrap.min.css"
import "bootstrap"
```

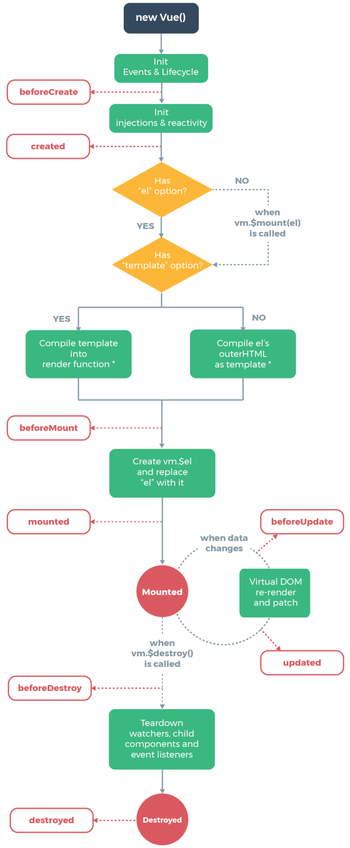
Vue.js

Lifecycle & Hook

Lifecycle Hook adalah bagian yang sangat penting untuk diketahui karena Vue akan menentukan pada step apa kode kita akan dieksekusi

source : https://docs.vuejs.id/v2/guide/instance.html

Vue.js



* template compilation is performed ahead-of-time if using
a build step, e.g. single-file components

Creation

Type lifecycle yang pertama kali dijalankan pada component, memungkinkan kita untuk menjalankan kode sesaat sebelum dan sesudah component mengupdate DOM

Creation :

- created

Created

Type lifecycle hook created dieksekusi oleh vue ketika data dan event telah selesai direnderr, tetapi template belum direnderr oleh vue


```
<div class="jumbotron text-center">

     <p>Lifecycle hook pada Vue js</p>

</div>
<div class="container">

  <div class="row">
     <div class=“col-md-12”>

       <div id="belajar-hook">{{pesan}}</div>

       <br/>
           <button @click="renderVirtualDOM()" >

           Render Virtual Dom{{rendered}}

      </button>

       </div>
  </div>
</div>
```


```
Deklarasi

Pesan: “”

Cara Mengecek

created() => {
  console.log(“log dari created”);
  console.log("pesan: " + pesan);
  console.log(document.getElementById(“belajar-hook"));
  console.log("");
});
```

Nilai property pesan yang ada di option data dapat dibaca melalui

```
console.log("pesan: " + pesan.value);
```

Akan tetapi,

```
console.log(document.getElementById(“belajar-hook"));
```

hanya akan menghasilkan null pada inspect Console karena elemen

```
<div id=“belajar-hook" >{{pesan}}</div>
```

tidak di-render oleh Vue.

penting : template belum di-render, cocok digunakan untuk logic pengambilan data dari API server.


```
console.log(document.getElementById(“belajar-hook"));
```

tidak menghasilkan null lagi pada inspect Console karena elemen

```
<div id=“belajar-hook" >{{pesan}}</div>
```

kini telah di-render oleh Vue.

penting : tidak cocok untuk logic pengambilan data langsung dari API server yang perlu pengolahan data lagi, sebab data sudah harus tersedia sebelum `template` di-render.

Created => unBeforeMount

Pada Composition atau Vue 3, created sudah tidak ada pada documentation composition tetapi secara konsep digantikan dungeon “unBeforeMount” dengan hasil yang sama

Deklarasi

```
const pesan = ref("");
```

Cara Mengecek

```
unBeforeMount(() => {
  console.log(“log dari un before mount");
  console.log("pesan: " + pesan.value);
  console.log(document.getElementById(“belajar-hook"));
  console.log("");
});
```

Mounting

Merupakan tipe lifecycle pada vue yang memungkinkan kita mengakses dom persis sebelum dan sesudah template di render, jangan digunakan untuk keperluan mengambil data atau event, karena template ini membutuhkan data tersebut sebelum ditampilkan

Mounting :

- Mounted

Mounted

Mounted dieksekusi oleh vue setelah template di render, kita dapat mengakases seluruh component, template, data, event, dan object global pada vue. Hal ini seperti kita menuliskan kode javascript didalam kode html body


```
<div class="jumbotron text-center">

     <p>Lifecycle hook pada Vue js</p>

</div>
<div class="container">

  <div class="row">
     <div class=“col-md-12”>

        <div id="belajar-hook">{{pesan}}</div>

       <br/>
           <button @click="renderVirtualDOM()" >

           Render Virtual Dom{{rendered}}

     </button>

      </div>
  </div>
</div>
```


```
Deklarasi

pssan: “”

Cara Mengecek

mounted() => {
  console.log(“log dari mounted”);
  console.log("pesan: " + pesan);
  console.log(document.getElementById(“belajar-hook"));
  console.log("");
});
```


```
console.log(document.getElementById(“belajar-hook"));
```

tidak menghasilkan null lagi pada inspect Console karena elemen

```
<div id=“belajar-hook" >{{pesan}}</div>
```

kini telah di-render oleh Vue.

penting : tidak cocok untuk logic pengambilan data dari API server, sebab data sudah harus tersedia sebelum `template` di-render.

Mounted => onMounted

Pada Composition atau Vue 3 mounted sudah tidak ada pada documentation tetapi secara konsep digantikan dengan “onMounted” dengan hasil yang sama

Deklarasi

```
const pesan = ref("");
```

Cara Mengecek

```
onMounted(() => {
  console.log(“log dari on mounted");
  console.log("pesan: " + pesan.value);
  console.log(document.getElementById(“belajar-hook"));
  console.log("");
});
```

Vue.js

Komunikasi antar component

Dalam VueJs, kita bisa memecah tampilan yang besar menjadi beberapa komponen kecil yang memiliki UI/ tampilan dan fungsinya sendiri-sendiri.

Namun dengan memecah tampilan menjadi beberapa komponen akan muncul masalah yaitu apabial terdapat data, props atau event yang musti di ekseskusi

Untuk Memudahkan kita memahami komunikasi antar component terkait data kita akan menggunakan istilah:

- Komunikasi parent ke child
- Komunikasi child ke parent

Komunikasi Parent ke Child

Untuk berkomunikasi dari parent component ke child component, kita bisa menggunakan **props**.

props memberikan kita jalan satu arah dari parent ke child. Tapi tidak bisa sebaliknya.

Untuk memberikan data ke child, parent bisa menambahkan argument di deklarasi child component.

Komunikasi Parent ke Child

Deklarasi

```
defineProps({
    data: {
        type: Object,
        default: {},
        require: false
    }
})
```

Cara Menggunakan

```
<detail-data :data="detail" ></detail-data>
```

Komunikasi Child ke Parent

Untuk berkomunikasi dari child ke parent, kita bisa menggunakan Kita akan menggunakan **emit** yang bisa gunakan untuk memanggil function di parent.

Komunikasi Parent ke Child

Deklarasi:

```
const emit = defineEmits([‘emitName'])
```

Cara Menggunakan

```
emit('emitName', “Argument/data”)
```

TASK

Noted

Akan diberikan 2 buah JSON dalam bentuk Array, dan buat suatu component tabel untuk bisa menampilkan kedua data tersebut, dan ada button show data untuk elihat data per baris dalar bentuk alert

No	Nama	Position	No Telp	Email	Action
1	Fauzan	Leader	fauzan@sinau.com	087126272716	Show Data
2	Albar	Vice Leader	albar@sinau.com	087126272716	Show Data
3	Bicky	Staff	bicky@sinau.com	087182832896	Show Data
4	Desi	Staff	desi@sinau.com	087182892839	Show Data
5	Anggi	Staff	anggi@sinau.com	087182838237	Show Data
6	Tommy	Staff	tommy@sinau.com	087182881238	Show Data