Rakamin Academy

Core Initiative

Project Based Internship

Javascript Props

Pengenalan mengenai Props pada Vue Js

Daftar Isi

A. Apa itu Props?

Props merupakan pendekatan paling mudah dan paling diajurkan ketika ingin melempar data dari satu komponen ke komponen lainnya. Pendekatan paling pas menggunakan Props adalah ketika kita ingin melempar data hanya kepada satu komponen lain.

Props merupakan one-way data flow sehingga hanya akan berjalan searah yakni dari parent ke child yang menerima Props tersebut, sementara child tidak bisa mengubah nilai Props yang dia terima dan tidak berhak pula melempar data ke parent nya.

Type dari props yang dapat anda gunakan antara lain :

- String
- Number
- Boolean
- Array
- Object
- Date
- Function
- Symbol

a. Inisiasi Props

Anda dapat menambahkan props pada component anda sebagai properti dari componen, lalu anda dapat merender dengan menggunakan cara menuliskan '{{ props }}' pada tag template.

```
<script>
export default {
  name: "user-detail",
  props: {
    name : String
  },
};
</script>
```

```
<template>
  <h1>{{name}}</h1>
</template>
```

b. Props Default

Props default merupakan data yang akan dirender ketika component anda gagal memuat props yang dikirimkan. Anda dapat memberi nilai default pada props anda dengan cara seperti berikut :

```
<script>
export default {
  name: "user-detail",
  props: {
    name : {
      type: String,
      default: "Default Value"
    }
  },
};
</script>
```

B. Mengirim Data Melalui Props

a. Parent ke Child

Anda dapat mengirimkan data dari parent ke child melalui props. Ada dua jenis props yang dapat dikirimkan, static props dan dynamic props. Untuk static props, anda tidak akan membuatnya fleksibel (tidak dapat diubah, kecuali dengan cara mengubahnya secara manual), sedangkan dynamic props nilainya dapat berubah ubah maka gunakanlah keduanya sesuai dengan kebutuhan anda.

```
<template>
  <ComponentSatu name="Sutejo"/>
  <ComponentSatu :name="data"/>
</template>

<script>
import ComponentSatu from './components/ComponentSatu'

export default {
  name: "App",
  components: {
    ComponentSatu
  },
  data(){
    return{
      data : "Agus"
    }
  }
};
</script>
```

```
<template>
  <h1>{{name}}</h1>
</template>

<script>
export default {
  name: "ComponentSatu",
  props: {
    name : {
      type: String,
      default: "Default Value"
    }
  },
};
</script>
```

Penulisan ":name" menggunakan (:) menandakan bahwa jenis props yang digunakan adalah dynamic props, sedangkan penulisan "name" yang tidak menggunakan (:) menandakan bahwa props yang digunakan adalah static props.

b. Child ke Parent

Jika sebelumya untuk mengirimkan data melalui props dari parent ke child terlihat lumayan sederhana, bagaimana jika kita ingin melakukan sebaliknya yaitu mengirimkan data dari child ke parent melalui props.

Untuk melakukan hal tersebut, anda dapat mengirimkan sebuah method atau function dari parent ke child, tujuan dari dikirimnya function untuk dijadikan sebagai wadah penerima data untuk nantinya digunakan oleh parent. Cara ini sering dikenal dengan event emit.

Sebagai contoh anda ingin membuat child berupa tombol yang dapat mengubah tampilan nama pada parent.

Untuk dapat melakukan hal tersebut, anda perlu menyiapkan method pada parent untuk dikirimkan ke child. Ketika method tersebut dijalankan, maka data person akan diubah mengikuti paramater yang diterima, paramater ini lah yang nantinya dikirmkan melalui child.

Parent

```
export default {
  name: "App",
  components: {
    ComponentSatu
  },
  data(){
    return{
      person : "Name"
    }
  },
  methods : {
    getDataFromChild(data){
      this.person = data
    }
  }
};
</script>
```

```
<template>
  <h1>{{person}}</h1>
  <ComponentSatu @sendDataFromChild="getDataFromChild"/>
</template>
```

*Props didahului dengan "@" yang menandakan method akan ada kaitannya dengan event, event akan kita pelajari pada materi selanjutnya.

Child

```
<script>
export default {
  name: "ComponentSatu",
  props: {
  },
  methods: {
    handleClick(){
      this.$emit('sendDataFromChild','Agus')
    },
    handleClick2() {
      this.$emit('sendDataFromChild', 'Joko')
    }
  }
};
</script>
```

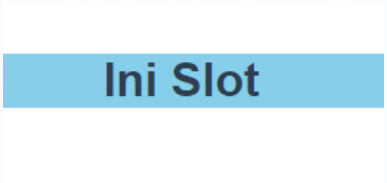
```
<template>
  <button @click="handleClick">Agus</button>
  <button @click="handleClick2">Joko</button>
</template>
```

C. Slots Component

Ketika kebutuhan program anda semakin besar, maka component tidak akan bisa menjadi sefleksibel dengan hanya menggunakan props saja. Solusi dari permasalahan tadi adalah dengan menggunakan slot, dimana parent dapat menentukan tag apa saja yang akan ada didalam childnya. Hal ini akan jauh lebih fleksibel dibandingkan hanya memanfaatkan props untuk membuat component menjadi fleksibel.

```
<template>
  <header>
    <slot />
  </header>
</template>
```

```
<template>
  <ComponentSatu>
    <h1>Ini Slot</h1>
  </ComponentSatu>
</template>
```



Anda juga dapat membuat dua slot dalam satu component, namun yang harus diperhatikan disetiap slot diperlukan nama untuk membantu vue mengidentifikasi slot mana yang akan disi. Anda dapat memberi nama slot dengan menambahkan attribute name pada tag slot, dan menggunakan attribut v-slot ketika menentukkan slot mana yang akan diisi.

```
<template>
  <header>
    <slot name="header"/>
  </header>

  <footer>
    <slot name="footer"/>
  </footer>
</template>
```

```
<template>

  <ComponentSatu>

    <template v-slot:header>
      <h1>Ini Header</h1>
    </template>

    <template v-slot:footer>
      <h1>Ini Footer</h1>
    </template>

  </ComponentSatu>

</template>
```

References

https://learn.microsoft.com/id-id/training/modules/vue-cli-components/

https://docs.vuejs.id/v2/guide/components-registration.html

https://variancode.com/vue-js/pembuatan-komponen-di-dalam-kompone

n-vue-js/

https://mfaridzia.medium.com/menerapkan-component-composition-di-vu

e-js-acb188b11606

https://ngide.net/belajar-vuejs-component