Contents

[Install 4](#_Toc161404563)

[Intoduction using Global Build 4](#_Toc161404564)

[Introduction Using the ES Module Build 5](#_Toc161404565)

[Reactive Example 5](#_Toc161404566)

[Vue binding Text(v-text, v-html) 6](#_Toc161404567)

[v-text 6](#_Toc161404568)

[v-html 7](#_Toc161404569)

[v-bind href 7](#_Toc161404570)

[v-bind src 7](#_Toc161404571)

[v-bind style, alt, title, class & id 8](#_Toc161404572)

[V-Model 10](#_Toc161404573)

[Computed Properties 11](#_Toc161404574)

[If-Else (v-if) 12](#_Toc161404575)

[v-show 13](#_Toc161404576)

[One line v-if 14](#_Toc161404577)

[Looping(v-for) 14](#_Toc161404578)

[Lopping Array 14](#_Toc161404579)

[Lopping Array of Object 15](#_Toc161404580)

[Event Handling & Methods 16](#_Toc161404581)

[Inline Method Click 16](#_Toc161404582)

[Click method 16](#_Toc161404583)

[KeyDown and KeyUp 17](#_Toc161404584)

[Event Prevent Default 17](#_Toc161404585)

[Watcher- Reactive Error Messages 18](#_Toc161404586)

[Watcher untuk validasi 18](#_Toc161404587)

[Contoh: Membuat TodoList 20](#_Toc161404588)

[Hooks 22](#_Toc161404589)

[Before Create & Created 22](#_Toc161404590)

[Before mount & mounted 23](#_Toc161404591)

[beforeUpdate & updated 24](#_Toc161404592)

[beforeUnmount &unmounted 24](#_Toc161404593)

[Get data from API 24](#_Toc161404594)

[Single File Component (SFC) 25](#_Toc161404595)

[SFC Adalah 25](#_Toc161404596)

[Install vue SFC 25](#_Toc161404597)

[Struktur File di SFC 26](#_Toc161404598)

[Index.html 26](#_Toc161404599)

[Main.js 26](#_Toc161404600)

[App.vue 26](#_Toc161404601)

[Component 27](#_Toc161404602)

[Create & Call Component 27](#_Toc161404603)

[Options API 27](#_Toc161404604)

[Props 28](#_Toc161404605)

[Emits 28](#_Toc161404606)

[Slot 30](#_Toc161404607)

[Router 31](#_Toc161404608)

[Install 31](#_Toc161404609)

[Setup 31](#_Toc161404610)

[Implementasi 32](#_Toc161404611)

[Dynamic Router 33](#_Toc161404612)

[Nested Route 34](#_Toc161404613)

[Same Route but different Param 35](#_Toc161404614)

[Options API vs Composition API 36](#_Toc161404615)

[Assigne object 36](#_Toc161404616)

[Watcher 37](#_Toc161404617)

[onMounted 37](#_Toc161404618)

[Style scooped 37](#_Toc161404619)

[Pinia – State Management (Store) 38](#_Toc161404620)

[Install Pinia 38](#_Toc161404621)

[Setup Pinia 38](#_Toc161404622)

[Create Store 38](#_Toc161404623)

[Contoh Todo List dengan Pinia 40](#_Toc161404624)

# Install

<https://vuejs.org/>

Bisa menggunakan CLI

npm create vue@latest

Bisa menggunakan CDN

<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>

Gunakan yg .prod untuk build production

<script src="https://unpkg.com/vue@3/dist/vue.global.prod.js"></script>

## Intoduction using Global Build

<head>

<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>

</head>

<body>

<div id="app">

    {{ message }}

</div>

    <script>

        const {createApp} = Vue

        createApp({

            data(){

                return{

                    message: 'Hello VueJs'

                }

            }

        }).mount('#app')

    </script>

</body>

Pertama buat const bernama createApp yg menandakan ini adalah aplikasi Vue

Kemudian buat structure dari createApp yg me-return data(), jangan lupa tambahkan .mount(‘#app’)

.mount disini berarti hasil return ini akan di pakai kedalam div yg id nya = app

Pada contoh diatas, kita me-return variable message, dan di baca dalam div menggunakan {{ message }}

## Introduction Using the ES Module Build

<head>

    <title>Belajar VueJs</title>

</head>

<body>

<div id="app">

    {{ message }}

</div>

    <script type="module">

        import { createApp} from 'https://unpkg.com/vue@3/dist/vue.esm-browser.js'

        createApp({

            data(){

                return{

                    message: 'Hello VueJs'

                }

            }

        }).mount('#app')

    </script>

</body>

Bila menggunakan ES Module, kita tidak perlu menginclude script pada head

Pada tag <script> tambahkan type=”module”, lalu import {createApp} dari alamat diatas

Untuk prod nya di

'https://unpkg.com/vue@3/dist/vue.esm-browser.prod.js'

## Reactive Example

<div id="app">

    <button @click="kurang()">-</button>

    {{ count }}

    <button @click="tambah()">+</button>

</div>

    <script type="module">

        import { createApp} from 'https://unpkg.com/vue@3/dist/vue.esm-browser.js'

        createApp({

            data(){

                return{

                    count: 0

                }

            },

            methods:{

                tambah(){

                    this.count++;

                },

                kurang(){

                    this.count--;

                }

            }

        }).mount('#app')

    </script>

# Vue binding Text(v-text, v-html)

## v-text

<div id="app">

    <div>

        <div v-text="message"></div>

        <div v-text="message2">

            random string

        </div>

    </div>

</div>

    <script type="module">

        import { createApp} from 'https://unpkg.com/vue@3/dist/vue.esm-browser.js'

        createApp({

            data(){

                return{

                    message: 'hallo vueJs',

                    message2: 'ini akan mereplace string diatas'

                }

            },

Hasil:

hallo vueJs

ini akan mereplace string diatas

v-text akan mereplace semua string yg ada di dalam div, dengan nilai yg ada di data()

## v-html

<div id="app">

    <div>

        <div v-html="message"></div>

    </div>

</div>

    <script type="module">

        import { createApp} from 'https://unpkg.com/vue@3/dist/vue.esm-browser.js'

        createApp({

            data(){

                return{

                    message: '<i> hallo vueJs dalam italic </i>'

                }

            },

Hasil:

*hallo vueJs dalam italic*

v-html akan mereplace semua string dalam div beserta tag html yg ada pada data()

## v-bind href

<a v-bind:href="url">Google</a>

    atau

<a :href="url">Google</a>

createApp({

        data(){

            return{

                url: 'https://google.com',

            }

        }

    }).mount('#app')

## v-bind src

<div>

<img v-bind:src='user.image' style="width:50px">

    atau

   <img :src='user.image' style="width:50px">

 </div>

createApp({

        data(){

            return{

                url: 'https://google.com',

                user:{

                    name: 'user1',

                    image: 'https://images.pexels.com/photos/1804035/pexels-photo-1804035.jpeg?auto=compress&cs=tinysrgb&w=600'

                }

            }

        }

## v-bind style, alt, title, class & id

<style>

    .textbiru {

        color: blue;

    }

    #textUnderLine {

        text-decoration: underline;

    }

</style>

<body>

<div id="app">

    <div :class="selectedClass" :id="selectedId">

        user: {{user.name}}

    </div>

    <div>

        <img

            v-bind:src='user.image'

            v-bind:style="{width: calculatedWidth}"

            v-bind:alt=" 'image of '+ user.name "

            v-bind:title="  'image of '+ user.name "

        >

        atau

        <img

            :src='user.image'

            :style="{width: calculatedWidth}"

            :alt=" 'image of '+ user.name "

            :title=" 'image of '+ user.name "

        >

    </div>

</div>

<script type="module">

    import { createApp } from 'https://unpkg.com/vue@3/dist/vue.esm-browser.js'

    createApp({

        data(){

            return{

                url: 'https://google.com',

                user:{

                    name: 'user1',

                    image: 'https://images.pexels.com/photos/1804035/pexels-photo-1804035.jpeg?auto=compress&cs=tinysrgb&w=600'

                },

                calculatedWidth: '400px',

                selectedClass: 'textbiru',

                selectedId: 'textUnderLine'

            }

        }

    }).mount('#app')

# V-Model

Untuk me-bind nilai di dalam form, agar di tempat lain, nilai nya bisa langsung ikut berubah

<div id="app">

    <div>

        <input type="text" v-model ="username">

    </div>

    <div>

        <textarea cols="30" rows="10" v-model ="address"></textarea>

    </div>

    <div>

        <input type="radio" value="L" v-model="gender">Laki-laki

        <input type="radio" value="P" v-model="gender">Perempuan

    </div>

    <div>

        <input type="checkbox" value="sepak bola" v-model="hobby"> Sepak Bola

        <input type="checkbox" value="basket" v-model="hobby"> Basket

        <input type="checkbox" value="voli" v-model="hobby"> Voli

    </div>

    <div>

        <select v-model="role">

            <option disabled value="">Choose one</option>

            <option value="admin">Admin</option>

            <option value="staff">Staff</option>

        </select>

    </div>

    <div>

        <p>

            username : {{username}}

            <br/>

            address : {{address}}

            <br/>

            gender : {{gender}}

            <br/>

            hobi:

            <span v-if="hobby.length > 0">

                <ul v-for="item in hobby">

                    <li>{{item}}</li>

                </ul>

            </span>

            <br/>

            role: {{role}}

        </p>

    </div>

</div>

<script type="module">

    import { createApp } from 'https://unpkg.com/vue@3/dist/vue.esm-browser.js'

    createApp({

        data(){

            return{

                username:'',

                address:'',

                gender:'',

                hobby:[],

                role:''

            }

        }

    }).mount('#app')

Disini kita membuat variabel bernama username, dengan nilai default kosong. Kemudian di buat input type text yg meng-bind variabel “username” tadi. Hasilnya ketika kita meng ketikan apapun nilai di input text, akan ter display di <p> bawahnya, dst

# Computed Properties

Memproses data untuk disajikan di view nya. Tidak butuh trigger(langsung berjalan)

<div id="app">

    <form action="">

        <input type="text" v-model="firstName">

        <input type="text" v-model="lastName">

    </form>

    <div>

        Full Name : {{fullName}}

    </div>

</div>

<script type="module">

    import { createApp } from 'https://unpkg.com/vue@3/dist/vue.esm-browser.js'

    createApp({

        data(){

            return{

                firstName: '',

                lastName: ''

            }

        },

        computed:{

            fullName(){

                return this.firstName + ' ' + this.lastName

            }

        }

    }).mount('#app')

\* ketika nilai v-model yg ada di dalam computed berubah, maka akan di jalankan kembali function computed nya

# If-Else (v-if)

<div id="app">

   <div v-if="tampil">

        {{ message }}

   </div>

   <div v-else>

        message false

   </div>

   <div v-if="nilai > 8">

        Nilai = A

   </div>

   <div v-else-if="nilai <= 8 && nilai > 6">

        Nilai = B

   </div>

   <div v-else>

        Nilai = C

   </div>

</div>

    <script type="module">

        import { createApp} from 'https://unpkg.com/vue@3/dist/vue.esm-browser.js'

        createApp({

            data(){

                return{

                   message: 'hello world',

                   tampil: true,

                   nilai: 5

                }

            },

        }).mount('#app')

## v-show

<div id="app">

    <div v-show="stateLampu">

        Lampu hidup

    </div>

</div>

    <script type="module">

        import { createApp} from 'https://unpkg.com/vue@3/dist/vue.esm-browser.js'

        createApp({

            data(){

                return{

                  stateLampu: false

                }

            },

        }).mount('#app')

Bedanya v-show dan v-if, pada v-show bila data nya false, maka akan ditambahkan

<div style="display: none;"> Lampu hidup </div>

Bila pada v-if, semua element nya tidak akan di munculkan semua

Kapan pakai v-show?

Lebih baik dipakai untuk data yg sering berubah-ubah /dynamic

<div id="app">

    <div v-show="stateLampu">

        Lampu hidup

    </div>

    <button @click="toogleLampu">Saklar Lampu</button>

</div>

    <script type="module">

        import { createApp} from 'https://unpkg.com/vue@3/dist/vue.esm-browser.js'

        createApp({

            data(){

                return{

                  stateLampu: false

                }

            },

            methods: {

                toogleLampu(){

                    this.stateLampu = !this.stateLampu

                }

            }

        }).mount('#app')

    </script>

## One line v-if

<span :class="{doneText : flagItem == true}">

Atau

<span :class="{doneText : flagItem}">

Artinya kita akan memakai class doneText jika nilai dari flagItem == true

# Looping(v-for)

<div id="app">

    <div v-for="angka in 10">

        {{ angka }}

    </div>

<ul v-for="angka in 10">

        <li>{{ angka }}</li>

    </ul>

</div>

Hasil:

1

2

3

4

5

6

7

8

9

10

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10

## Lopping Array

<div id="app">

    <div v-for="(value, index) in fruits">

        {{index}} - {{value}} <button @click="hapusBuah(index)">Delete</button>

    </div>

</div>

<script type="module">

    import { createApp } from 'https://unpkg.com/vue@3/dist/vue.esm-browser.js'

    createApp({

        data(){

            return{

                fruits: ['apple', 'orange', 'melon', 'lemon', 'papaya']

            }

        },

        methods: {

            hapusBuah(index) {

                this.fruits.splice(index,1)

            }

        }

    }).mount('#app')

</script>

## Lopping Array of Object

<div id="app">

    <div v-for="(value, index) in footballPlayers">

        {{index}} - {{value.name}} | {{value.position}}

    </div>

</div>

<script type="module">

    import { createApp } from 'https://unpkg.com/vue@3/dist/vue.esm-browser.js'

    createApp({

        data(){

            return{

                footballPlayers:[

                    {name:'messi', position: 'FW'},

                    {name:'CR7', position: 'WF'},

                    {name:'casilas', position: 'GK'},

                    {name:'ramos', position: 'DF'},

                    {name:'modric', position: 'MF'},

                ]

            }

        },

    }).mount('#app')

</script>

# Event Handling & Methods

## Inline Method Click

<div id="app">

    <button v-on:click="count--">-</button>

        <span> {{count}} </span>

    <button v-on:click="count++">+</button>

</div>

<script type="module">

    import { createApp } from 'https://unpkg.com/vue@3/dist/vue.esm-browser.js'

    createApp({

        data(){

            return{

                count: 0

            }

        },

Bentuk singkat dari v-on:click adalah @click

<div id="app">

    <button @click="count--">-</button>

        <span> {{count}} </span>

    <button @click="count++">+</button>

</div>

## Click method

<div id="app">

    <button @click="kurang()">-</button>

        <span> {{count}} </span>

    <button @click="tambah()">+</button>

    <button @click="sayHello('cara fajar')">Say Hello</button>

</div>

<script type="module">

    import { createApp } from 'https://unpkg.com/vue@3/dist/vue.esm-browser.js'

    createApp({

        data(){

            return{

                count: 0

            }

        },

        methods: {

            kurang(){

                this.count--

            },

            tambah(){

                this.count++

            },

            sayHello(nama){

                alert('Hallo '+ nama)

            }

        }

    }).mount('#app')

</script>

## KeyDown and KeyUp

<input type="text" v-on:keydown.enter="sayHello('cara fajar')">

<br/>

<input type="text" v-on:keydown.space="sayHello('cara fajar')">

Keydown.enter = Event ketika kita menekan tombol enter, maka akan menjalankan method sayHello

Keydown.space = Event ketika kita menekan tombol spasi, maka akan menjalankan method sayHello

Bentuk singkatnya:

<input type="text" @keydown.enter="sayHello('cara fajar')">

<br/>

<input type="text" @keydown.space="sayHello('cara fajar')">

## Event Prevent Default

<form method="post" action="whatever.html" v-on:submit.prevent>

     <input type="text">

     <input type="submit" value="send">

</form>

Prevent default disini, menghilangkan sifat default dari form, yakni ketika di button submit akan me-refresh halaman. Dengan menambahkan submit.prevent, ketika di button submit, maka halaman tidak akan di refresh

Bentuk singkat:

 <form method="post" action="whatever.html" @submit.prevent>

# Watcher- Reactive Error Messages

Menangkap perubahan yang ada pada element

<div id="app">

    <form action="">

        <input type="text" v-model="firstName">

    </form>

</div>

<script type="module">

    import { createApp } from 'https://unpkg.com/vue@3/dist/vue.esm-browser.js'

    createApp({

        data(){

            return {

                firstName: ''

            }

        },

        watch: {

            firstName(newValue, oldValue) {

                console.log('oldValue : ' + oldValue)

                console.log('newValue : ' + newValue)

            }

        }

    }).mount('#app')

</script>

Disini kita punya input text dengan model firstName, firstName kita watch, dengan membuat watch dengan nama yg sama dengan v-model nya.

Parameter pada function watch, sesuai urutan:

Param1 = new value dari inputan user

Param2 = old value dari inputan user sebelumnya

## Watcher untuk validasi

<div id="app">

    <form action="" @submit-prevent="">

        Username:

        <input type="text" v-model="userName">

        <label for="username" v-if="userNameErr">{{ errMSg }}</label>

        <br/>

        Password:

        <input type="password" v-model="password">

        <label for="password" v-if="passwordErr">{{ errMSgPassword }}</label>

        <br/>

        Retype Password:

        <input type="password" v-model="passwordRetype">

        <label for="password" v-if="reTypeErr">{{ errMSgRetype }}</label>

        <br/>

        <input type="submit" value="Register">

    </form>

</div>

<script type="module">

    import { createApp } from 'https://unpkg.com/vue@3/dist/vue.esm-browser.js'

    createApp({

        data(){

            return {

                userName: '',

                password: '',

                passwordRetype: '',

                userNameErr: false,

                passwordErr:false,

                reTypeErr:false,

                errMSg:'',

                errMSgPassword: '',

                errMSgRetype:''

            }

        },

        watch: {

            userName(value){

                if(value.length < 8)

                {

                    this.userNameErr = true

                    this.errMSg = 'minimal username 8 char'

                } else {

                    this.userNameErr = false

                    this.errMSg = ''

                }

            },

            password(value){

                if(value.length < 8)

                {

                    this.passwordErr = true

                    this.errMSgPassword = 'minimal password 8 char'

                } else {

                    this.passwordErr = false

                    this.errMSgPassword = ''

                }

            },

            passwordRetype(value){

                if(value != this.password){

                    this.reTypeErr = true

                    this.errMSgRetype = 'password tidak sama'

                } else {

                    this.reTypeErr = false

                    this.errMSgRetype = ''

                }

            }

        }

    }).mount('#app')

</script>

# Contoh: Membuat TodoList

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Belajar VueJs</title>

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">

</head>

<style>

    .doneText{

        text-decoration: line-through;

        color: red;

    }

</style>

<body>

<div id="app">

  <div class="container">

    <h1 class="mt-3 text-center">TO DO LIST</h1>

    <div class="row justify-content-center">

        <div class="col-9 mt-5">

            <div class="row">

                <div class="col-2">

                    <input type="text" class="form-control" placeholder="start" v-model="startTime">

                </div>

                <div class="col-2">

                    <input type="text" class="form-control" placeholder="end" v-model="endTime">

                </div>

                <div class="col-6">

                    <input type="text" class="form-control" placeholder="new activity" v-model="activity">

                </div>

                <div class="col-2">

                    <button class="btn btn-primary form-control" @click="addToDo">Add</button>

                </div>

            </div>

        </div>

    </div>

    <div class="row justify-content-center">

        <div class="col-7 mt-5">

            <div v-for="(item, index) in todoList" class="p-2" style="border-bottom: solid 1px #ddd;">

                <button type="button" class="btn btn-outline-danger me-3" @click="deleteTodo(index)">delete</button>

                <button type="button" class="btn btn-success me-3" @click="changeStatus(index)">done</button>

                <span :class="{doneText : item.status}">

                    {{item.start}} - {{item.end}} : {{item.text}}

                </span>

            </div>

        </div>

    </div>

  </div>

</div>

<script type="module">

    import { createApp } from 'https://unpkg.com/vue@3/dist/vue.esm-browser.js'

    createApp({

        data(){

            return {

               todoList : [

                    {text: 'belajar HTML', start: '07:00', end: '08:00', status: false},

                    {text: 'belajar css', start: '08:00', end: '09:00', status: false},

                    {text: 'belajar JS', start: '09:00', end: '10:00', status: false}

               ],

               startTime: '',

               endTime:'',

               activity:''

            }

        },

        methods: {

            addToDo(){

                let newItem = {

                    text: this.activity,

                    start: this.startTime,

                    end: this.endTime,

                    status: false

                }

                this.todoList.push(newItem)

            },

            deleteTodo(index){

                this.todoList.splice(index, 1)

            },

            changeStatus(index){

                this.todoList[index].status = true

            }

        }

    }).mount('#app')

</script>

</body>

</html>

# Hooks

## Before Create & Created

    createApp({

        data(){

            return {

               message: 'hello world'

            }

        },

        beforeCreate(){

            console.log(`before create  ${this.message}`)

        },

        created(){

            console.log(`created ${this.message}`)

        }

    }).mount('#app')

Hasil:

before create undefined

created hello world

Beforecreate() dijalankan sebelum aplikasi mengenali isi dari data()

Created() ketika aplikasi sudahs selesai meng-create data()

## Before mount & mounted

<body>

<div id="app">

    <input type="text" value="whatever" ref="namaRef">

</div>

<script type="module">

    import { createApp } from 'https://unpkg.com/vue@3/dist/vue.esm-browser.js'

    createApp({

        data(){

            return {

               message: 'hello world'

            }

        },

        beforeMount(){

            console.log(`before : ${this.$refs.namaRef.value}`) // error

        },

        mounted(){

            console.log(`mounted : ${this.$refs.namaRef.value}`) // whatever

        }

    }).mount('#app')

</script>

beforeMount() menjalankan isi function sebelum view DOM dibuat, oleh karena itu jika kita get value dari input text diatas, akan return error

mounted() menjalankan function setelah view DOM selesai dibuat

## beforeUpdate & updated

<div id="app">

    <input type="text" :value="count" ref="hitungan">

    <button @click="count++">Add</button>

</div>

<script type="module">

    import { createApp } from 'https://unpkg.com/vue@3/dist/vue.esm-browser.js'

    createApp({

        data(){

            return {

               message: 'hello world',

               count: 0

            }

        },

        beforeUpdate(){

            console.log(`before udpate : ${this.$refs.hitungan.value}`) // 0

        },

        updated(){

            console.log('updated : '+ this.$refs.hitungan.value) // 1

        }

    }).mount('#app')

</script>

beforeUpdate() dijalankan sebelum ada perubahan dari sisi DOM View nya

updated() dijalankan setelah ada perubahan DOM / View

## beforeUnmount &unmounted

menggunakan component, coming soon

# Get data from API

<script type="module">

        import { createApp } from 'https://unpkg.com/vue@3/dist/vue.esm-browser.js'

        createApp({

            data(){

                return {

                }

            },

            methods: {

                async getListPokemon(){

                 let response = await fetch("https://pokeapi.co/api/v2/pokemon");

                 let data = await response.json();

                 console.log(data);

                }

            },

            mounted(){

                this.getListPokemon()

            }

    }).mount('#app')

# Single File Component (SFC)

## SFC Adalah

SFC adalah Single File Component artinya setiap komponen di buat dalam 1 file saja

## Install vue SFC

Arahkan ke direktori folder project, lalu npm create vue@latest

✔ Project name: … <your-project-name>

✔ Add TypeScript? … No / Yes

✔ Add JSX Support? … No / Yes

✔ Add Vue Router for Single Page Application development? … No / Yes

✔ Add Pinia for state management? … No / Yes

✔ Add Vitest for Unit testing? … No / Yes

✔ Add an End-to-End Testing Solution? … No / Cypress / Playwright

✔ Add ESLint for code quality? … No / Yes

✔ Add Prettier for code formatting? … No / Yes

Scaffolding project in ./<your-project-name>...

Done.

Tuliskan nama project

lalu pilih No semua(karena tutorial)

setelah selesai, masuk ke folder nama project dan jalankan

npm install

dan

npm run dev

alamat url: <http://127.0.0.1:5173/>

## Struktur File di SFC

### Index.html

<div id="app"></div>

    <script type="module" src="/src/main.js"></script>

Lokasi id=”app” nya ada disini

Memanggil script js type=”modul” dan memanggil main.js

### Main.js

import './assets/main.css'

import { createApp } from 'vue'

import App from './App.vue'

createApp(App).mount('#app')

memanggil css dari assets/main.css

import file dari APP.vue kemudian di mout ke div id=”app”

### App.vue

Pada bagian atas kita bisa melihat banyak mengimport komponen

<script setup>

import HelloWorld from './components/HelloWorld.vue'

import TheWelcome from './components/TheWelcome.vue'

</script>

<template>

  <header>

    <img alt="Vue logo" class="logo" src="./assets/logo.svg" width="125" height="125" />

    <div class="wrapper">

      <HelloWorld msg="You did it!" />

    </div>

  </header>

  <main>

    <TheWelcome />

  </main>

</template>

# Component

## Create & Call Component

Lokasi: \src\components\Student.vue

Tiap komponen wajib terdapat 1 tag <template> </template>

App.vue

<template>

  <Student />

</template>

<script setup>

  import Student from './components/Student.vue'

</script>

\*posisi <script setup> diatas atau di bawah sama aja

Student.Vue

<template>Hallo student</template>

## Options API

Di vue ada 2 jenis API, composition API dan Options API. Secara default vue SFC menggunkan composition API (<script setup>). Di pembahasan sebelumnya menggunakan Options API. Sehingga cara kita import agak sedikit diubah

App.vue

<template>

  <Student />

</template>

<script>

  import Student from './components/Student.vue'

  export default {

    components: {

      Student

    }

  }

</script>

Hanya menggunakan tag <script> Tanpa ‘setup’ dan setelah import Student, tambah export default

## Props

Dapat mengirim data ke child component nya

App.vue

<template>

  <Student studentName="Andy"/> <br>

  <Student studentName="Budi"/> <br>

  <Student studentName="Elly"/>

</template>

<script>

  import Student from './components/Student.vue'

  export default {

    components: {

      Student

    }

  }

</script>

Student.vue

<template>Hallo student {{ studentName }}</template>

<script>

    export default {

        props:{

            studentName : String

        }

    }

</script>

Di Student.vue kita daftarkan juga nama props nya

## Emits

Kebalikan dari props, mengirim value dari child component ke parent component

Student.vue

<template>

    Hallo student {{ studentName }}

    <button @click="sendName">Show</button>

</template>

<script>

    export default {

        props:{

            studentName : String

        },

        emits:['responseName'],

        methods:{

            sendName(){

                this.$emit('responseName', this.studentName)

            }

        }

    }

</script>

Buat emits dalam bentuk array. Saat kita klik tombol show, akan di jalankan methods sendName yg isi nya mengirim studentName dalam emit responseName

App.vue

<template>

  <Student @responseName="(value)=> selectedStudent=value" studentName="Andy"/> <br>

  <Student @responseName="showData" studentName="Budi"/> <br>

  <Student @responseName="showData" studentName="Elly"/>

  <p>

    Student yg di pilih = {{ selectedStudent }}

  </p>

</template>

<script>

  import Student from './components/Student.vue'

  export default {

    components: {

      Student

    },

    data(){

      return{

        selectedStudent:'',

        testData:''

    }

  },

  methods:{

    showData(value){

      this.selectedStudent = value

    }

  }

}

</script>

Pada tag <Student> kita tambahkan @responseName (nama emitsnya) dan apa yg akan dilakukan. Ada 2 cara:

* + Arrow function “(value)=> selectedStudent=value”
  + Panggil method bernama showData

## Slot

Informasi tambahan yg di kirim parent component ke children component.

Sifat nya tidak wajib / tambahan.

Bisa di beri nilai default jika parent tidak mengirim apa2

Harus menggunakan tag buka dan tag tutup (<Student> </Student>)

App.vue

<template>

  <Student @responseName="(value)=> selectedStudent=value" studentName="Andy">

    ini adalah nilai slot Parent

  </Student>

  <br>

  <Student @responseName="showData" studentName="Budi"/> <br>

  <Student @responseName="showData" studentName="Elly"/> <br>

Student.vue

<template>

    <slot>default msg</slot>

    <br>

    Hallo student {{ studentName }}

    <button @click="sendName">Show</button>

    <br>

</template>

Hasil:

ini adalah nilai slot Parent  
Hallo student Andy Show  
  
default msg  
Hallo student Budi Show  
  
default msg  
Hallo student Elly Show

# Router

## Install

Saat pertama npm install pilih yes

✔ Add Vue Router for Single Page Application development? … No / Yes

Atau

$npm install vue-router@4

## Setup

Buat folder dan file index.js di \src\router\index.js

import { createRouter, createWebHashHistory } from "vue-router";

import DashboardView from "../views/Dashboard.vue"

import AboutView from "../views/About.vue"

const routes = [

    { path:'', component: DashboardView },

    { path:'/about', component: AboutView }

]

const router = createRouter({

    history: createWebHistory(),

    routes

})

export default router

main.js

import { createApp } from 'vue'

import App from './App.vue'

import router from './router'

createApp(App).use(router).mount('#app')

\*tambahkan import router dan .use(router)

## Implementasi

App.vue

<template>

  <div class="menu">

    <Navbar/>

  </div>

  <RouterView />

</template>

<script>

    import Navbar from "./components/Navbar.vue"

    export default {

      components: {

        Navbar

      }

    }

</script>

Navbar.vue

<template>

    <RouterLink to="/">Dashboard</RouterLink>

    <RouterLink to="about">About</RouterLink>

</template>

Router Link disini sama dengan <a href=””>

<RouterLink :to="'user/'+val.params">{{ val.name }}</RouterLink>

Bisa juga pakai :to jika ingin di concat

RouterView tempat isi dari component akan di render, sehingga aplikasi tetap di halaman yg sama/ single page component

# Dynamic Router

Router/index.js

import UserView from "../views/User.vue"

import UserDetailView from "../views/UserDetail.vue"

const routes = [

    { path:'', component: DashboardView },

    { path:'/about', component: AboutView },

    { path:'/user', component: UserView },

    { path:'/user/:id', component: UserDetailView }

]

const router = createRouter({

    history: createWebHistory(),

    routes

})

export default router

disini kita akan membuat halaman user detail dengan mengirimkan param id di url

User.vue

<template>

    <h1>Ini adalah halaman USer</h1>

    <h2>List User</h2>

    <ul>

        <li v-for="val in users">

            <RouterLink :to="'user/'+val.params">{{ val.name }}</RouterLink>

        </li>

    </ul>

</template>

<script>

    export default {

        data() {

            return {

                users:[

                    {params: 'meg', name: 'Meg'},

                    {params: 'ben', name: 'Ben'},

                    {params: 'greg', name: 'Greg'},

                ]

            }

        }

    }

</script>

UserDetail.vue

<template>

    Halaman detail dari {{ $route.params.id }}

</template>

\*kenapa .id karena di router/index.js kita pakai :id

# Nested Route

Router/index.js

const routes = [

    { path:'', component: DashboardView },

    { path:'/about', component: AboutView },

    { path:'/user', component: UserView },

    //regular routes format

    // { path:'/user/:id', component: UserDetailView },

    // { path:'/user/:id/profile', component: UserProfileView },

    // { path:'/user/:id/posts', component: UserPostView }

    // nested routes format

    {

        path:'/user/:id',

        component: UserIndexView,

        children: [

            {

                path:'',

                component: UserDetailView

            },

            {

                path:'/user/:id/posts',

                component: UserPostView

            },

            {

                path:'/user/:id/profile',

                component: UserProfileView

            },

        ]

    }

Disini kita perlu 1 component tambahan (UserIndexView)

UserIndex.vue

<template>

    <RouterView/>

</template>

UserPost.vue

<template>

    Halaman POST dari user {{ $route.params.id }}

</template>

UserProfile.vue

<template>

    halaman Profile dari user: {{ $route.params.id }}

</template>

# Same Route but different Param

Contoh pada project pokedex-sfc

Kita punya route(index.js)

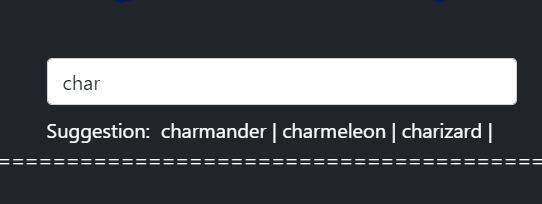
{

      path:'/pokemon/:name',

      component: DetailPokemon,

}

Dan kita memiliki navigasi seperti di bawah:



Bila user pilih charmander, maka url = http://localhost:5173/pokemon/charmander

Bila user pilih charmeleon, maka url = http://localhost:5173/pokemon/ charmeleon

Bila user pilih charizard, maka url = http://localhost:5173/pokemon/ charizard

Ketiga link tersebut memanggil path yg sama ('/pokemon/:name')

Route vue akan menganggap tidak ada perubahan halaman. Agar halaman tetap bisa reactive/ berubah detail nya, maka kita harus watch perubahan param nya, di halaman component

DetailPokemon.vue

watch(() => route.params.name, (newId, oldId) => {

getPokemon() //call api

})

# Options API vs Composition API

Biasanya options API hanya menggunakan tag <script> sedangkan pada composition API mengunakan <script setup>

Composition API

<template>

<button @click="methodComposition()">Alert</button>

    {{ profile }}

    {{ age }}

    {{ computedAge }}

</template>

<script setup>

import {ref, reactive, onMounted, computed} from 'vue'

const profile = reactive({name:'sadam', age:'17'}) //only declares reactive state for objects.

const age = ref(12) // allows us to declare reactive state for primitives and objects

function methodComposition(){

    alert('wasd')

}

const computedAge = computed(() => {

    console.log(age.value);

    console.log(profile.age);

    return profile.age\*2

})

## Assigne object

<script setup>

let selectedPokemon = reactive({})

async function getPokemon(){

        const name = route.params.name

        let response = await fetch("https://pokeapi.co/api/v2/pokemon/" + name);

        let data = await response.json();

        Object.assign(selectedPokemon, data)

    }

</script>

## Watcher

<script setup>

import {ref, watch } from 'vue'

let search = ref('')

watch(search, (currentValue, oldValue) => {

     console.log(currentValue);

     console.log(oldValue);

});

</script>

Contoh lain watcher

watch(() => route.params.name, (newId, oldId) => {

        getPokemon()

    })

## onMounted

<script setup>

import {ref, reactive, onMounted, computed} from 'vue'

onMounted(() => {

  getListPokemon()

})

</script>

# Style scooped

<style scoped>

    .menu{

        background-color: #eee;

        line-height: 2.5;

        padding: 10px;

    }

</style>

Artinya style ini hanya berlaku di halaman component tempat style ini di buat

# Pinia – State Management (Store)

Misal kita punya counter yg menghitung sampai angka 5, bila menggunakan store, ketika pindah halaman, maka angka di counter tetap pada angka 5

Dianjurkan install extension vue js devtools

## Install Pinia

<https://pinia.vuejs.org/>

$npm install pinia

## Setup Pinia

Src/main.js

Tambahkan

import { createPinia } from 'pinia'

const pinia = createPinia()

app.use(router).use(pinia)

## Create Store

Create new folder ‘stores’ in src/stores dan file di dalam nya bernama todoStore.js

todoStore.js

import { defineStore } from 'pinia'

export const useTodoStore = defineStore('todo', {

})

Penamaan const sebaiknya menggunakan use<nama>Store (diapit)

‘todo’ disini sebaiknya id unik di 1 aplikasi

Sama seperti vue api, ada 2 jenis Store yakni option Store dan composition Store. Disarankan pakai option store

import { defineStore } from 'pinia'

export const useTodoStore = defineStore('todo', {

    state: () => ({ count: 0, name: 'Eduardo' }),

    getters: {

        doubleCount: (state) => state.count \* 2,

    },

    actions: {

        increment() {

            this.count++

        },

    },

})

State: sama dengan data di vue reguler

Getters: sama dengan computed di vue reguler

Actions: sama dengan methods di vue reguler

TodoView.vue

<template>

    <h1>Halaman todolist</h1>

    count = {{ todoStore.count }}

    <br>

    name = {{ todoStore.name }}

    <br>

    doubleCount = {{ todoStore.doubleCount }}

    <br><br>

    <button @click="todoStore.increment">Add</button>

</template>

<script setup>

import { useTodoStore } from '../stores/todoStore'

const todoStore = useTodoStore()

</script>

# Contoh Todo List dengan Pinia

todoStore.js

import { defineStore } from 'pinia'

export const useTodoStore = defineStore('todo', {

    state: () => ({

        todoList: [

            {'name' : 'Belajar HTML', isDone: false},

            {'name' : 'Belajar CSS', isDone: false},

            {'name' : 'Belajar JS', isDone: false},

            {'name' : 'Belajar PHP', isDone: false},

        ]

    }),

    getters: {

       showAll(state){

            return state.todoList

       },

       doneOnly(state){

            return state.todoList.filter(item => item.isDone == true)

       },

       unDoneOnly(state){

            return state.todoList.filter(item => item.isDone == false)

       }

    },

    actions: {

        setAsDone(name){

            this.todoList.find(item => item.name == name).isDone = true

        },

        setAsUnDone(name){

            this.todoList.find(item => item.name == name).isDone = false

        },

        addTodo(data){

            let exist = this.todoList.filter(item=>item.name == data).length

            if(exist) {

                alert('Already Exist')

                return

            }

            this.todoList.push(

                {name : data, isDone: false}

            )

        }

    },

})

TodoView.vue

<template>

    <h1>Halaman todolist</h1>

    <form @submit.prevent="todoStore.addTodo(newTodo)">

        <input type='text' v-model="newTodo" placeholder="add new todo ....">

        <input type="submit" value="Add">

    </form>

    <div style="width:500px; padding-inline-start: 40px;">

        <h3 style="text-align: center;">My Todo List</h3>

    </div>

    <div class="getters-button">

        <button @click="show = 'all'">Show All</button>

        <button @click="show = 'done only'">Done Only</button>

        <button @click="show = 'undone only'">Undone Only</button>

    </div>

    <!-- show all -->

    <div v-if="show == 'all'">

        <ul>

            <li v-for="list in todoStore.todoList">

                <span>{{ list.name }}</span>

                <span>

                    <button v-if="list.isDone == false" @click="todoStore.setAsDone(list.name)">set as Done</button>

                    <button v-if="list.isDone" @click="todoStore.setAsUnDone(list.name)">set as Un-Done</button>

                </span>

            </li>

        </ul>

    </div>

    <!-- done only -->

    <div v-if="show == 'done only'">

        <ul>

            <li v-for="list in todoStore.doneOnly">

                <span>{{ list.name }}</span>

                <span>

                    <button v-if="list.isDone == false" @click="todoStore.setAsDone(list.name)">set as Done</button>

                    <button v-if="list.isDone" @click="todoStore.setAsUnDone(list.name)">set as Un-Done</button>

                </span>

            </li>

        </ul>

    </div>

    <!-- undone only -->

    <div v-if="show == 'undone only'">

        <ul>

            <li v-for="list in todoStore.unDoneOnly">

                <span>{{ list.name }}</span>

                <span>

                    <button v-if="list.isDone == false" @click="todoStore.setAsDone(list.name)">set as Done</button>

                    <button v-if="list.isDone" @click="todoStore.setAsUnDone(list.name)">set as Un-Done</button>

                </span>

            </li>

        </ul>

    </div>

</template>

<style scooped>

    form{

        width:500px;

        padding-inline-start: 40px;

        margin-bottom: 40px;

    }

    form input {

        padding:10px

    }

    form input:first-child{

        width: 80%;

        margin: 0px 10px;

    }

    ul{

        list-style: none;

        width: 500px;

    }

    li{

        border: solid 1px;

        margin:10px;

        padding: 10px;

        display: flex;

        justify-content: space-between;

    }

    .getters-button{

        display: flex;

        justify-content: center;

        gap: 10px;

        width:500px;

        padding-inline-start: 40px;

    }

</style>

<script setup>

import { useTodoStore } from '../stores/todoStore'

import {ref, reactive, onMounted, computed} from 'vue'

const todoStore = useTodoStore()

const newTodo= ref()

let show= ref('all')

</script>

Dengan option api

<script>

    import { useTodoStore } from '../stores/todoStore'

    export default{

        setup(){

            const todoStore = useTodoStore()

            return {todoStore}

        }

    }

</script>