**Vue JS**

Installation

VUE Commands

**Install Vue JS**

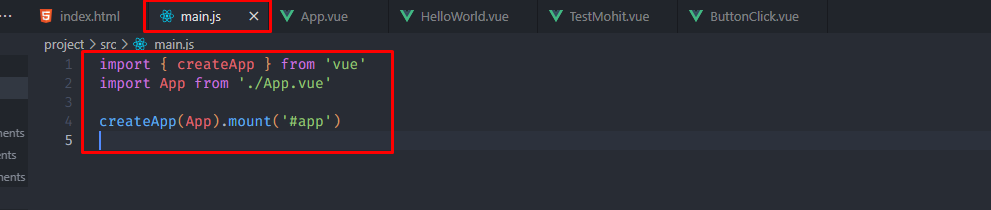
npm install -g @vue/cli

vue create project-name

cd project-name

npm run serve

Main.js

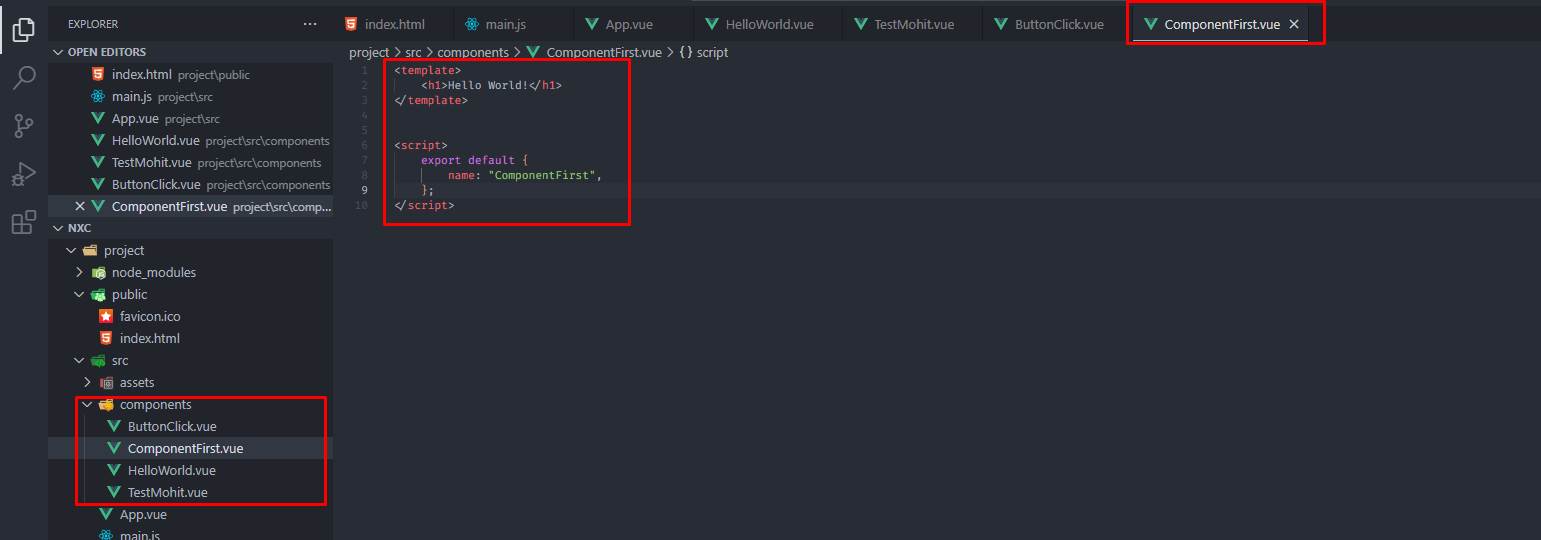


import { createApp } from 'vue'

import App from './App.vue'

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

Create Simple Component



<template>

<h1>Hello World!</h1>

</template>

<script>

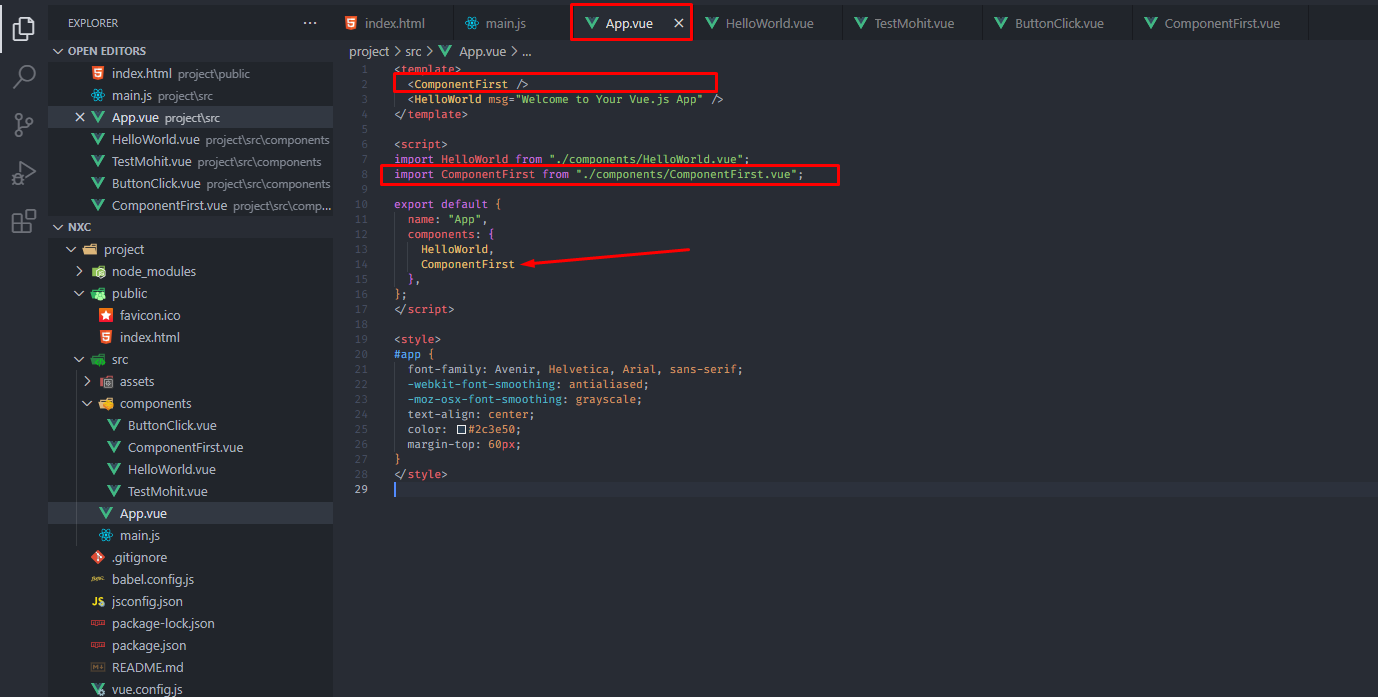
export default {"

name: "ComponentFirst,

};

</script>

Call Component in App.Vue



<template>

<ComponentFirst />

<HelloWorld msg="Welcome to Your Vue.js App" />

</template>

<script>

import HelloWorld from "./components/HelloWorld.vue";

import ComponentFirst from "./components/ComponentFirst.vue";

export default {

name: "App",

components: {

HelloWorld,

ComponentFirst

},

};

</script>

<style>

#app {

font-family: Avenir, Helvetica, Arial, sans-serif;

-webkit-font-smoothing: antialiased;

-moz-osx-font-smoothing: grayscale;

text-align: center;

color: #2c3e50;

margin-top: 60px;

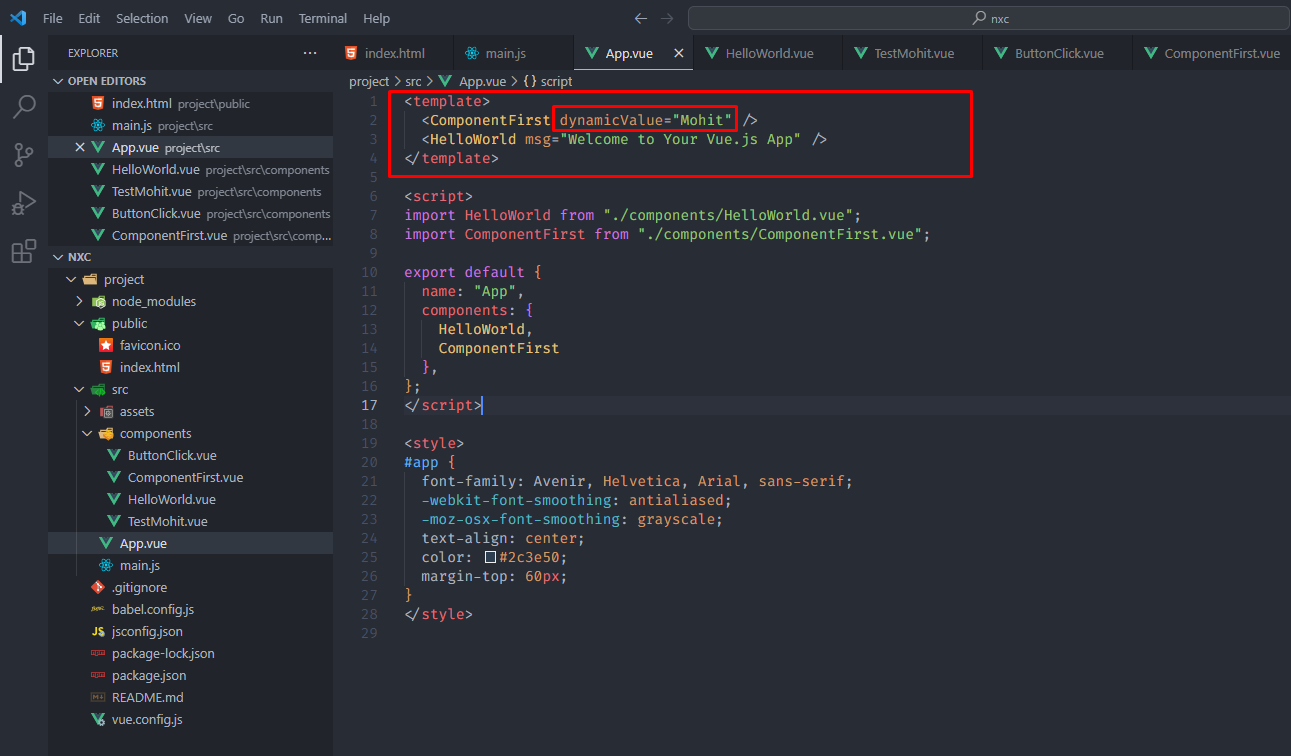
}

</style>

Props

Props for transfer data between components

Dynamically pass props & access props in component



<template>

<ComponentFirst dynamicValue="Mohit" />

<HelloWorld msg="Welcome to Your Vue.js App" />

</template>

<script>

import HelloWorld from "./components/HelloWorld.vue";

import ComponentFirst from "./components/ComponentFirst.vue";

export default {

name: "App",

components: {

HelloWorld,

ComponentFirst

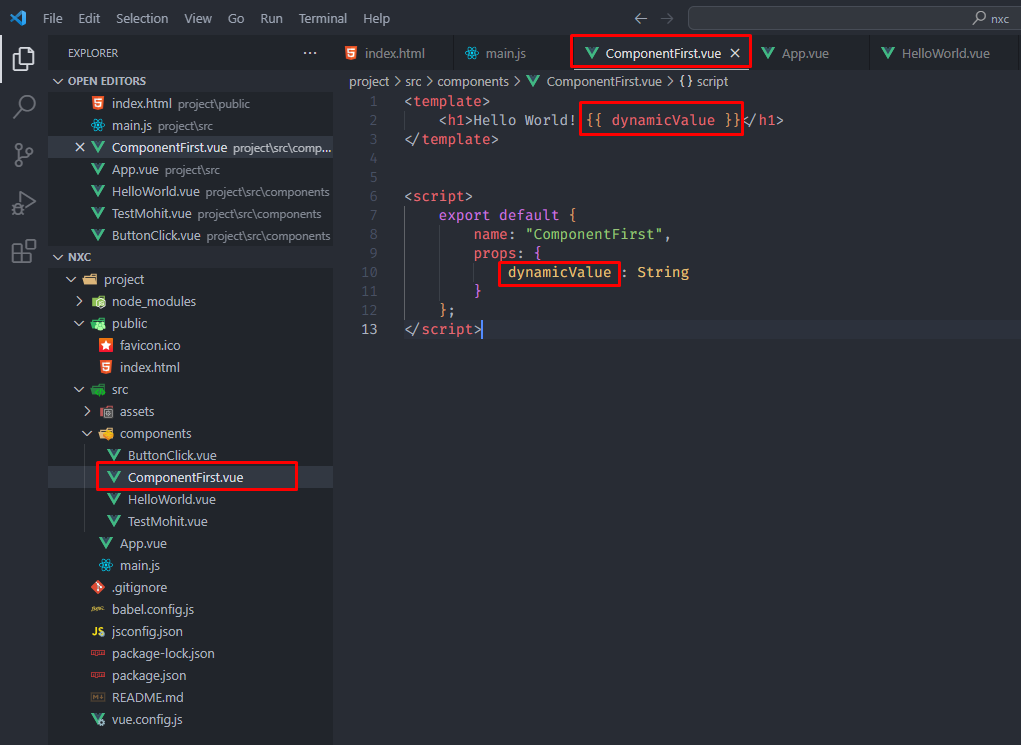
},

};

</script>

Access props in component

.



<template>

<h1>Hello World! {{ dynamicValue }}</h1>

</template>

<script>

export default {

name: "ComponentFirst",

props: {

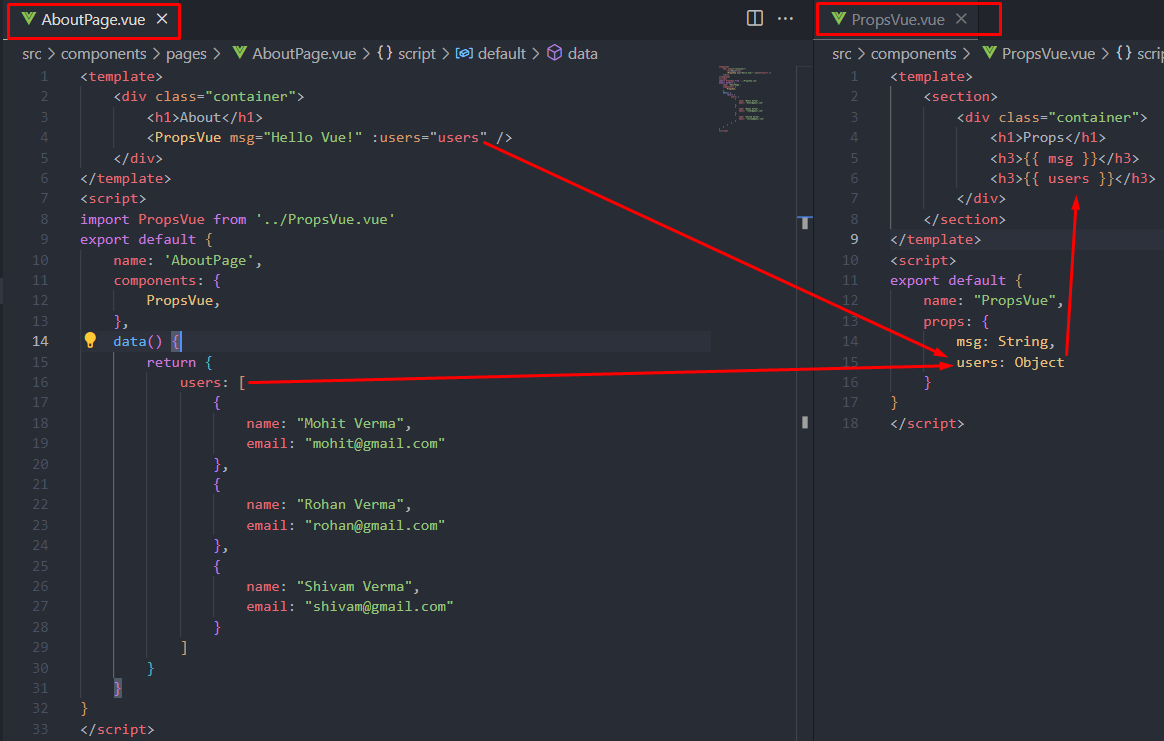
dynamicValue : String

}

};

</script>

Props Pass Object from parent to child



**AboutPage.vue 👇**

<template>

<div class="container">

<h1>About</h1>

<PropsVue msg="Hello Vue!" :users="users" />

</div>

</template>

<script>

import PropsVue from '../PropsVue.vue'

export default {

name: 'AboutPage',

components: {

PropsVue,

},

data() {

return {

users: [

{

name: "Mohit Verma",

email: "mohit@gmail.com"

},

{

name: "Rohan Verma",

email: "rohan@gmail.com"

},

{

name: "Shivam Verma",

email: "shivam@gmail.com"

}

]

}

}

}

</script>

**PropsVue.vue 👇**

<template>

<section>

<div class="container">

<h1>Props</h1>

<h3>{{ msg }}</h3>

<h3>{{ users }}</h3>

</div>

</section>

</template>

<script>

export default {

name: "PropsVue",

props: {

msg: String,

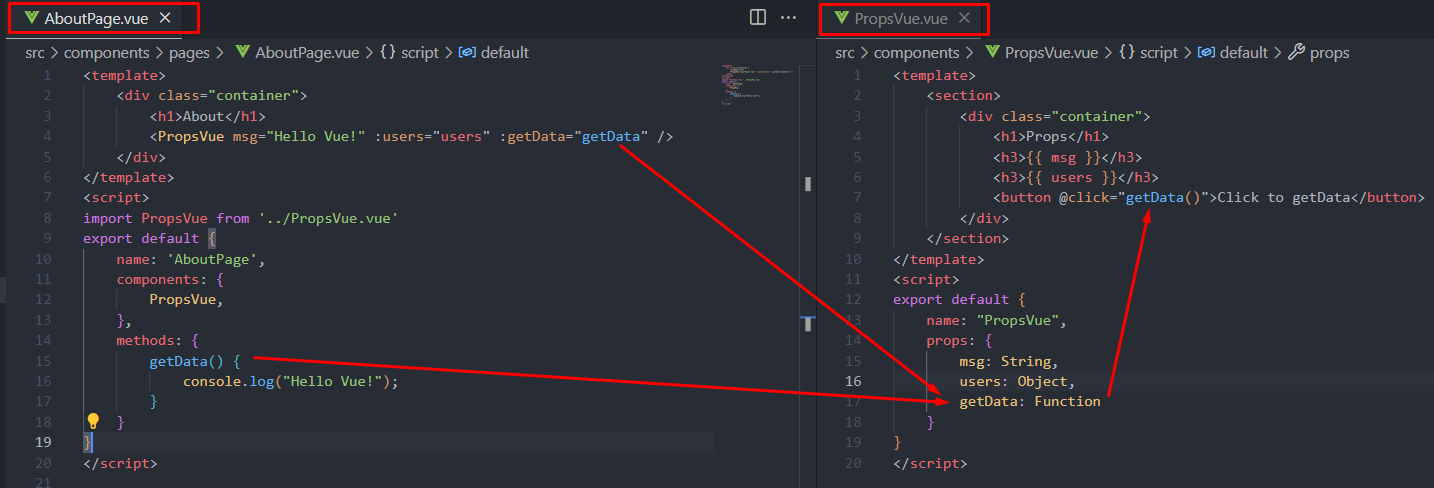
users: Object

}

}

</script>

Props pass function from parent to child



**AboutPage.vue 👇**

**<template>**

**<div class="container">**

**<h1>About</h1>**

**<PropsVue msg="Hello Vue!" :users="users" :getData="getData" />**

**</div>**

**</template>**

**<script>**

**import PropsVue from '../PropsVue.vue'**

**export default {**

**name: 'AboutPage',**

**components: {**

**PropsVue,**

**},**

**methods: {**

**getData() {**

**console.log("Hello Vue!");**

**}**

**}**

**}**

**</script>**

**PropsVue.vue 👇**

**<template>**

**<section>**

**<div class="container">**

**<h1>Props</h1>**

**<h3>{{ msg }}</h3>**

**<h3>{{ users }}</h3>**

**<button @click="getData()">Click to getData</button>**

**</div>**

**</section>**

**</template>**

**<script>**

**export default {**

**name: "PropsVue",**

**props: {**

**msg: String,**

**users: Object,**

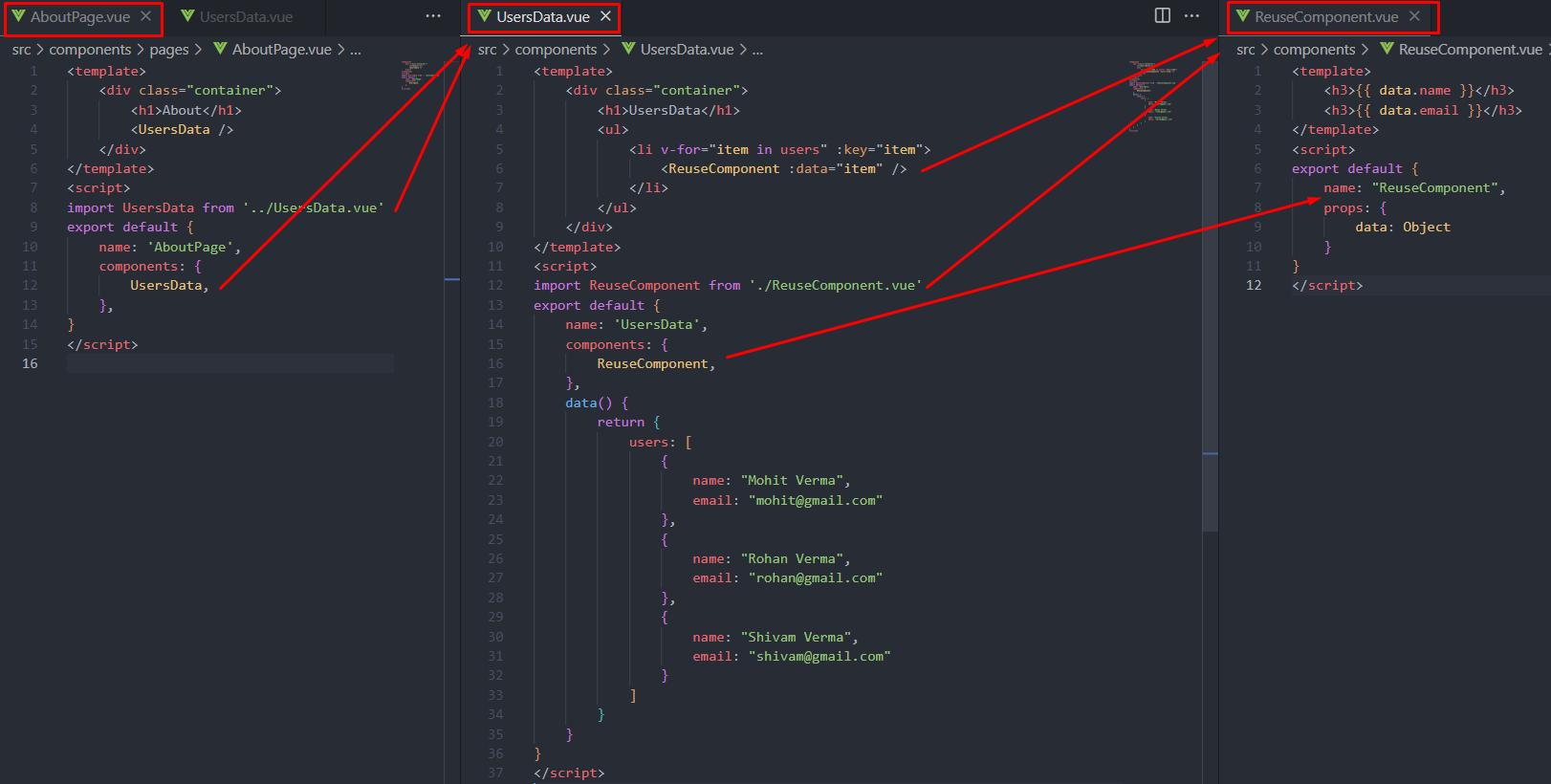
**getData: Function**

**}**

**}**

**</script>**

Props Reusable Component



**AboutPage.vue 👇**

**<template>**

**<div class="container">**

**<h1>About</h1>**

**<UsersData />**

**</div>**

**</template>**

**<script>**

**import UsersData from '../UsersData.vue'**

**export default {**

**name: 'AboutPage',**

**components: {**

**UsersData,**

**},**

**}**

**</script>**

**UsersData.vue 👇**

**<template>**

**<div class="container">**

**<h1>UsersData</h1>**

**<ul>**

**<li v-for="item in users" :key="item">**

**<ReuseComponent :data="item" />**

**</li>**

**</ul>**

**</div>**

**</template>**

**<script>**

**import ReuseComponent from './ReuseComponent.vue'**

**export default {**

**name: 'UsersData',**

**components: {**

**ReuseComponent,**

**},**

**data() {**

**return {**

**users: [**

**{**

**name: "Mohit Verma",**

**email: "mohit@gmail.com"**

**},**

**{**

**name: "Rohan Verma",**

**email: "rohan@gmail.com"**

**},**

**{**

**name: "Shivam Verma",**

**email: "shivam@gmail.com"**

**}**

**]**

**}**

**}**

**}**

**</script>**

**ReuseComponent.vue 👇**

**<template>**

**<h3>{{ data.name }}</h3>**

**<h3>{{ data.email }}</h3>**

**</template>**

**<script>**

**export default {**

**name: "ReuseComponent",**

**props: {**

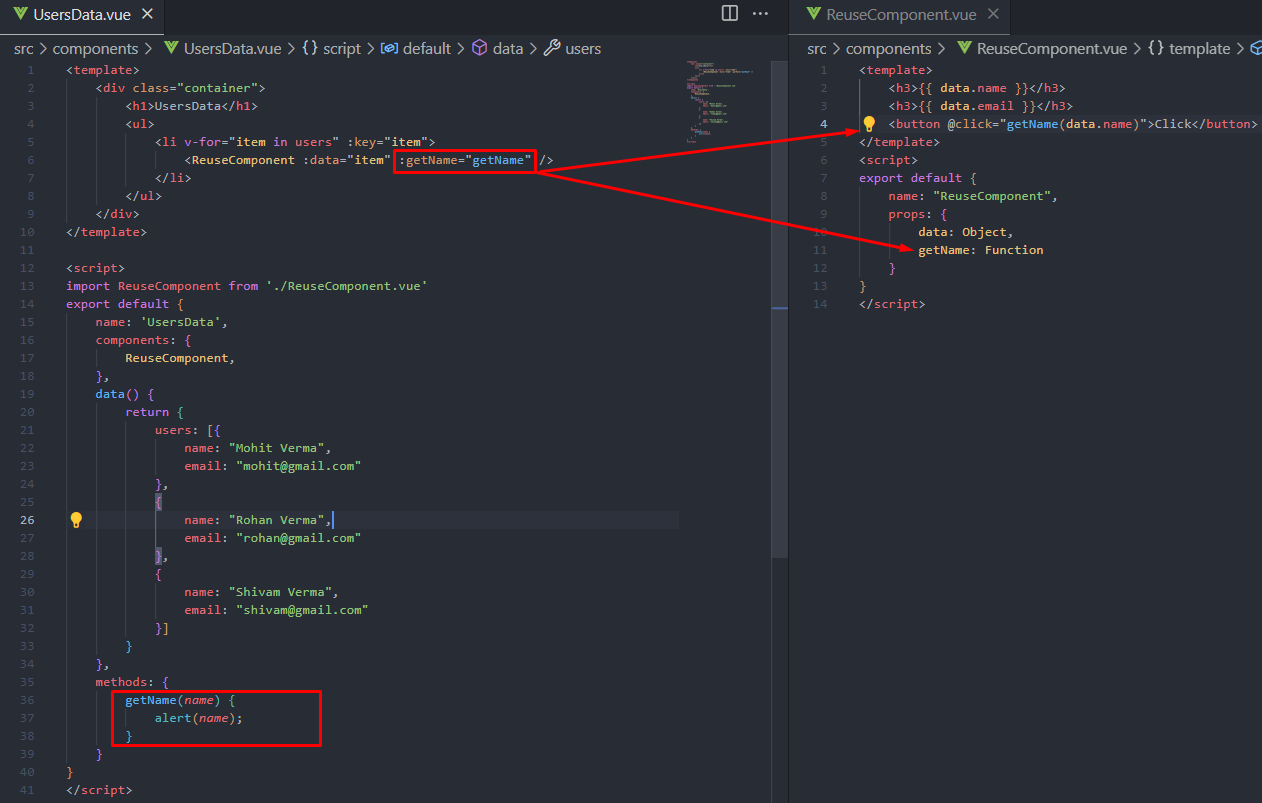
**data: Object**

**}**

**}**

**</script>**

Pass a function from the **parent component** to the **child component**, where the function is defined in the parent.👇



**UsersData.vue 👇**

**<template>**

**<div class="container">**

**<h1>UsersData</h1>**

**<ul>**

**<li v-for="item in users" :key="item">**

**<ReuseComponent :data="item" :getName="getName" />**

**</li>**

**</ul>**

**</div>**

**</template>**

**<script>**

**import ReuseComponent from './ReuseComponent.vue'**

**export default {**

**name: 'UsersData',**

**components: {**

**ReuseComponent,**

**},**

**data() {**

**return {**

**users: [{**

**name: "Mohit Verma",**

**email: "mohit@gmail.com"**

**},**

**{**

**name: "Rohan Verma",**

**email: "rohan@gmail.com"**

**},**

**{**

**name: "Shivam Verma",**

**email: "shivam@gmail.com"**

**}]**

**}**

**},**

**methods: {**

**getName(*name*) {**

**alert(*name*);**

**}**

**}**

**}**

**</script>**

**ReuseComponent.vue 👇**

**<template>**

**<h3>{{ data.name }}</h3>**

**<h3>{{ data.email }}</h3>**

**<button @click="getName(data.name)">Click</button>**

**</template>**

**<script>**

**export default {**

**name: "ReuseComponent",**

**props: {**

**data: Object,**

**getName: Function**

**}**

**}**

**</script>**

Props pass from child to parent

**ChildComp.vue 👇**

<template>

<h1>ChildComp</h1>

<button class="btn btn-primary" @click="getData(username)">Parent to Child Data</button>

</template>

<script>

export default {

name: 'ChildComp',

data() {

return {

username: "Mohit"

}

},

props: {

getData: Function

}

}

</script>

**ParentComp.vue 👇**

**<template>**

**<ChildComp :getData="getDataFromParent" />**

**<h1>ParentComp</h1>**

**</template>**

**<script>**

**import ChildComp from './ChildComp.vue'**

**export default {**

**name: 'ParentComp',**

**components: {**

**ChildComp,**

**},**

**methods: {**

**getDataFromParent(*name*) {**

**console.log(*name*);**

**}**

**}**

**}**

**</script>**

Scoped Css

When a <style> tag has the scoped attribute, its CSS will apply to elements of the current component only.

<style scoped>

.example {

color: red;

}

</style>

<template>

<div class="example">hi</div>

</template>

Interpolation & Data

The most basic form of data binding is text interpolation using the "Mustache" syntax (double curly braces):

<span>Message: {{ msg }}</span>

The double mustaches interpret the data as plain text, not HTML. In order to output real HTML, you will need to use the v-html directive:

<p>Using text interpolation: {{ rawHtml }}</p>

<p>Using v-html directive: <span v-html="rawHtml"></span></p>

<template>

<h1>{{ name }}</h1>

<h1>{{ name.length }}</h1>

*<!-- reassign value -->*

<h1>{{ (name = "Mohit") }}</h1>

*<!-- check condition isEqual return true / false == -->*

<h1>{{ name == "Mohit" }}</h1>

<h1>{{ email }}</h1>

<h1>{{ getData() }}</h1>

<h1>{{ getMessage("Msg") }}</h1>

</template>

<script>

export default {

name: "TestMohit",

props: {

dynamicText: String,

},

data() {

return {

name: "Mohit",

email: "mohitverma.techmind@gmail.com",

getData: function () {

*// console.log(this.name);*

return "This is comes from function inside data() ";

},

};

},

};

</script>

Methods & Params

<template>

*<!-- <button @click="count++">Add 1</button> -->*

*<!-- <button v-on:click="count++">Add 1</button> -->*

<button v-on:click="count++">Add 1</button>

<p>Count is: {{ count }}</p>

<button @click="greet">Greet</button>

<button @click="greet1('btn 1 click')">Button - 1</button>

<button @click="greet2('btn 2 click')">Button - 2</button>

<button @click="greet">Greet</button>

<div class="box" v-on:mousemove="mouseMoveTest">

MouseMove

<p id="clientX"></p>

</div>

*<!-- <div class="box" @mousemove="mouseMoveTest">MouseMove</div> -->*

</template>

<script>

export default {

name: "ButtonClick",

data() {

return {

count: 0,

};

},

methods: {

greet() {

alert(`Hello Greet!`);

},

greet1(*data*) {

alert(`${*data*}`);

},

greet2(*data*) {

alert(`${*data*}`);

},

mouseMoveTest(*event*) {

document.getElementById("clientX").innerHTML = *event*.x;

},

getDetails() {

return {

age: 18,

subject: this.subj,

};

},

},

};

</script>

*<!--Styling for mousemove box-->*

<style>

.box{

position: relative;

width: 200px;

height: 200px;

background-color: yellow;

}

</style>

Two Way Binding

Simple with text field

<template>

*<!-- This is Vue -->*

<div class="form\_wrap">

<h2>Form Data Two Way Binding</h2>

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

<div class="output">{{ output }}</div>

</div>

</template>

<script>

*// This is Model*

export default{

name: "TwoWayBinding",

data() {

return{

output : "Hi"

}

}

}

</script>

With name & email fields

<template>

*<!-- This is Vue -->*

<div class="form\_wrap">

<h2>Form Data Two Way Binding</h2>

<form>

<div class="input\_wrap">

<input type="text" v-model="name" placeholder="Name">

</div>

<div class="input\_wrap">

<input type="email" v-model="email" placeholder="Email">

</div>

<div class="submit\_wrap">

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

</div>

<div class="output">

{{ `Name = ${name} Email = ${email}` }}

</div>

</form>

</div>

</template>

<script>

*// This is Model*

export default{

name: "TwoWayBinding",

data() {

return{

name: "",

email: ""

}

}

}

</script>

With name & email fields & add Class conditional base Dynamically using Variable

<template>

*<!-- This is Vue -->*

<div class="form\_wrap">

<h2>Form Data Two Way Binding</h2>

<form>

<div class="input\_wrap">

<input type="text" v-model="name" placeholder="Name">

</div>

<div class="input\_wrap">

<input type="email" v-model="email" placeholder="Email">

</div>

<div class="submit\_wrap">

<input type="submit" value="Submit" v-on:click="getData">

</div>

<div class="output"

:class="{ 'showClassAdded': show }"

>

<table>

<tr>

<th>Name</th>

<th>Email</th>

</tr>

<tr>

<td>{{name}}</td>

<td>{{email}}</td>

</tr>

</table>

</div>

</form>

</div>

</template>

<script>

*// This is Model*

export default{

name: "TwoWayBinding",

data() {

return{

name: "",

email: "",

show: false,

}

},

methods: {

getData(*event*){

*event*.preventDefault();

if ((this.name != "") && (this.email != "")) {

alert("success");

this.show = true;

}else {

alert("please fill all fields");

this.show = false;

}

}

}

}

</script>

Radio & Checkboxes

<template>

*<!-- This is Vue -->*

<div class="form\_wrap">

<h2>Form Data Two Way Binding</h2>

<form>

<div class="input\_wrap">

<label>

<input type="checkbox" v-model="technology" value="HTML">

HTML

</label>

<label>

<input type="checkbox" v-model="technology" value="CSS">

CSS

</label>

<label>

<input type="checkbox" v-model="technology" value="JS">

JS

</label>

</div>

<div class="input\_wrap">

<label>

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

Male

</label>

<label>

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

Female

</label>

</div>

<div class="submit\_wrap">

<input type="submit" value="Submit" v-on:click="getData">

</div>

<div class="output"

:class="{ 'showClassAdded': show }"

>

<table>

<tr>

<th>Technology</th>

<th>Gender</th>

</tr>

<tr>

<td>{{technology}}</td>

<td>{{gender}}</td>

</tr>

</table>

</div>

</form>

</div>

</template>

<script>

*// This is Model*

export default{

name: "TwoWayBinding",

data() {

return{

technology:[],

gender: "",

show: false

}

},

methods: {

getData(*event*){

*event*.preventDefault();

if ((this.technology != "")) {

alert("success");

this.show = true;

}else {

alert("please fill all fields");

this.show = false;

}

}

}

}

</script>

Add Bootstrap - 5

Run this command in project folder

npm install bootstrap@5.3.0

After Install bootstrap -5 module  
Go to main.js and import bootstrap using below

import { createApp } from 'vue'

import App from './App.vue'

import 'bootstrap/dist/css/bootstrap.css'; *// Import Bootstrap styles*

import 'bootstrap'; *// Import Bootstrap JavaScript components*

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

Flag Icon library module in Vue Js

npm install flag-icon-css

import "flag-icon-css/css/flag-icons.css"

<span class="flag-icon flag-icon-cn"></span>

Create Router

Run this command in project folder

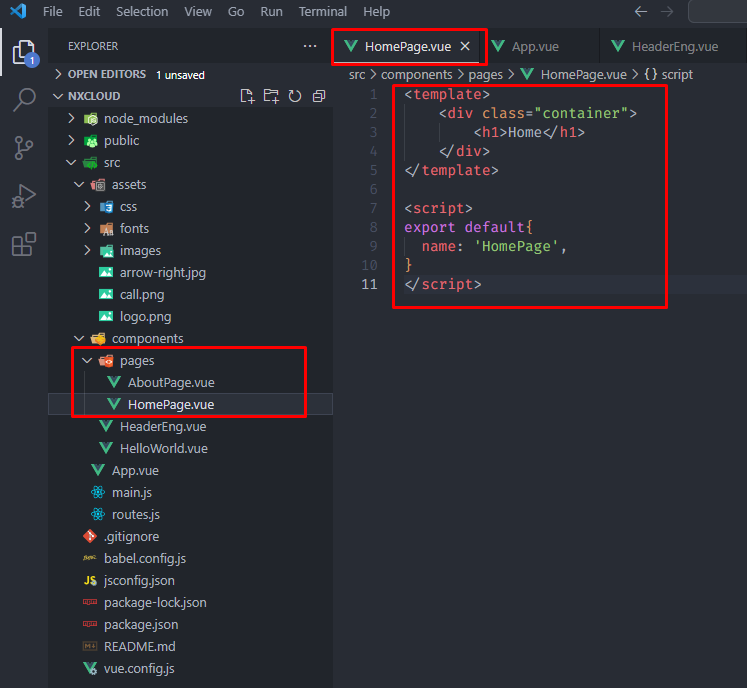
npm i vue-router@next

After that Create **routes.js** in inside **project folder > src** folder

**Note** : You need to create **routes.js file in src folder**

& import website’s pages in **routes.js**

Create Pages before adding in **routes.js mentioned below 👇**



<template>

<div class="container">

<h1>Home</h1>

</div>

</template>

<script>

export default{

name: 'HomePage',

}

</script>

In **routes.js** add this code 👇

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

import Home from "./components/pages/HomePage.vue";

import AboutPage from "./components/pages/AboutPage.vue";

*// component: () => import('./components/Home.vue'), //for lazy load*

// const Home = () => import("./components/pages/HomePage.vue"); *//for lazy load*

const routes = [

{

name : 'Home',

path : '/',

component : Home

},

{

name : 'About',

path : '/about',

component : AboutPage

},

]

const router = createRouter({

history : createWebHistory(),

routes

});

export default router;

After that open **main.js** and import router code add this code 👇

import { createApp } from 'vue'

import App from './App.vue'

import router from './routes.js'

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

Open **YourHeaderFile.vue** and create menu items links like this 👇

<template>

<header>

<div class="container">

<router-link to="/"> Home </router-link>

<router-link to="/about"> About </router-link>

</div>

</header>

<router-view></router-view>

</template>

<script>

export default {

name: 'HeaderEnglish'

}

</script>

**Dynamic Page Title**

In **routes.js** add this code 👇

**import { createWebHistory, createRouter } from "vue-router";**

**import Home from "./components/pages/HomePage.vue";**

**const routes = [**

**{**

**name : 'Home',**

**path : '/',**

**component : Home,**

**meta: {**

**title: 'Home Page' *// Set the title for this route***

**}**

**}**

**]**

**const router = createRouter({**

**history : createWebHistory(),**

**routes**

**});**

**export default router;**

In **main.js** add this code 👇

import { createApp } from 'vue'

import App from './App.vue'

import router from './routes.js'

*// for dynamic title*

router.beforeEach((*to*, *from*, *next*) => {

document.title = *to*.meta.title || 'Default Title';

next();

});

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

**Change Favicon**

In **index.html** add this code 👇

<link rel="icon" href="<%= BASE\_URL %>favicon.png">

**Note** : Make sure your favicon file place in the **public** directory of your project.

Add Og:Image thumbnail

In **index.html** add this code 👇

<link rel="icon" href="<%= BASE\_URL %>favicon.png">

<title><%= htmlWebpackPlugin.options.title %></title>

<meta property="og:image" content="/og-img.jpg">

**Note** : Make sure your og-img.jpg file place in the **public** directory of your project.

Useful Commands

npm install axios

npm install cookies-js

**import where you want to use these**

import axios from 'axios';

import Cookies from 'cookies-js';

Open project in VS directly by cmd 👇

code .

**Step for create a Build for your project**

npm run build

After that dist folder will created direct in your project folder

Like below

\projects\vue\nxc\live\nxcloud\dist

**Create local URL for your build**

To install http-server, you can use npm, the package manager for Node.js. Here's how you can install http-server globally using npm:

npm install -g http-server

http-server

**Or another way**

npm install -g serve

serve -s dist

Swiper Slider

**Installation**

npm i swiper

**Basic Slider**

<template>

<swiper

:slides-per-view="3"

:space-between="50"

@swiper="onSwiper"

@slideChange="onSlideChange"

>

<swiper-slide>Slide 1</swiper-slide>

<swiper-slide>Slide 2</swiper-slide>

<swiper-slide>Slide 3</swiper-slide>

...

</swiper>

</template>

<script>

*// Import Swiper Vue.js components*

import { Swiper, SwiperSlide } from 'swiper/vue';

*// Import Swiper styles*

import 'swiper/css';

export default {

components: {

Swiper,

SwiperSlide,

},

setup() {

const onSwiper = (*swiper*) => {

console.log(swiper);

};

const onSlideChange = () => {

console.log('slide change');

};

return {

onSwiper,

onSlideChange,

};

},

};

</script>

**Autoplay & Navigation**

**<swiper**

**:modules="modules"**

**:slides-per-view="1"**

**:space-between="50"**

**:autoplay="{ delay: 4000, disableOnInteraction: false }"**

**navigation**

**:pagination="{ clickable: true }"**

**@swiper="onSwiper"**

**@slideChange="onSlideChange"**

**>**

**<swiper-slide class="item"></swiper-slide>**

**<swiper-slide class="item"></swiper-slide>**

**</swiper>**

**<script>**

***// import Swiper core and required modules***

**import { Navigation, Pagination, Scrollbar, A11y, Autoplay } from 'swiper/modules';**

***// Import Swiper Vue.js components***

**import { Swiper, SwiperSlide } from 'swiper/vue';**

***// Import Swiper styles***

**import 'swiper/css';**

**import 'swiper/css/navigation';**

**import 'swiper/css/pagination';**

**import 'swiper/css/scrollbar';**

**import 'swiper/css/autoplay';**

***// Import Swiper styles***

**export default {**

**components: {**

**Swiper,**

**SwiperSlide,**

**},**

**setup() {**

**const onSwiper = (*swiper*) => {**

**console.log(*swiper*);**

**};**

**const onSlideChange = () => {**

**console.log('slide change');**

**};**

**return {**

**onSwiper,**

**onSlideChange,**

**modules: [Navigation, Pagination, Scrollbar, A11y, Autoplay],**

**};**

**},**

**};**

**</script>**

Data AOS Animation on scroll

Add AOS module in your project by this command

npm install aos --save

**Importing**

Import AOS into the app.vue file

<script>

import HeaderEng from './components/HeaderEng.vue'

import FooterEng from './components/FooterEng.vue'

*// Aos*

import AOS from 'aos'

import 'aos/dist/aos.css'

export default {

name: 'App',

components: {

HeaderEng,

FooterEng

},

mounted() {

AOS.init(

{

*// Global settings:*

disable: false,

startEvent: 'DOMContentLoaded',

initClassName: 'aos-init',

animatedClassName: 'aos-animate',

useClassNames: false,

disableMutationObserver: false,

debounceDelay: 50,

throttleDelay: 99,

offset: 120,

delay: 0,

duration: 2000,

easing: 'ease-in-out-back',

once: false,

mirror: false,

anchorPlacement: 'top-bottom'

}

)

},

}

</script>

For animation attributes

<https://github.com/michalsnik/aos#predefined-options>

<https://egghead.io/blog/how-to-use-the-animate-on-scroll-aos-library-in-vue>

Events in Vue

Add on scroll event in vue

<template>

<HeaderEng />

<FooterEng />

</template>

<script>

import HeaderEng from './components/HeaderEng.vue'

import FooterEng from './components/FooterEng.vue'

*// Aos*

import 'aos/dist/aos.css'

export default {

name: 'App',

components: {

HeaderEng,

FooterEng

},

created () {

window.addEventListener('scroll', this.handleScroll);

},

unmounted () {

window.removeEventListener('scroll', this.handleScroll);

},

methods: {

*// Any code to be executed when the window is scrolled*

handleScroll (*event*) {

console.log(*event*);

}

}

}

</script>

Add event when **App** is loaded

methods: {

getUnits() {

console.log("App is loaded");

},

},

beforeMount() {

this.getUnits();

},

Vue Extensions

Vue Language Features (Volar)

Vue-format

If Else Condition

<template>

<section>

<div class="container">

<h1 v-if="showData">Hello If</h1>

<h1 v-else>Hello Else</h1>

<button @click="changeCondition">Toggle Condition</button>

</div>

</section>

</template>

<script>

export default {

name: "IfElse",

data() {

return {

showData: false

}

},

methods: {

changeCondition() {

if (this.showData == true) {

this.showData = false;

} else {

this.showData = true;

}

}

}

}

</script>

Shorthand toggle condition

<template>

<section>

<div class="container">

<h1 v-if="showData">Hello If</h1>

<h1 v-else>Hello Else</h1>

<button @click="showData = !showData">Toggle Condition</button>

</div>

</section>

</template>

<script>

export default {

name: "IfElse",

data() {

return {

showData: false

}

}

}

</script>

For Loop

Simple for loop (Array)

<template>

<section>

<div class="container">

<h1>For Loop</h1>

<ul v-for="item in skills" :key="item">

<li>{{ item }}</li>

</ul>

</div>

</section>

</template>

<script>

export default {

name: "ForLoop",

data() {

return {

skills: ["HTML", "CSS", "JAVASCRIPT"]

}

}

}

</script>

For Loop with Array of an Object

Simple for loop (Array of an object)

<template>

<section>

<div class="container">

<h1>For Loop Array of an Object</h1>

<ul>

<li v-for="item in users" :key="item">

<h3>Name : {{ item.name }}</h3>

<h3>Email : {{ item.email }}</h3>

</li>

</ul>

</div>

</section>

</template>

<script>

export default {

name: "ForLoop",

data() {

return {

users: [

{

name: "Mohit Verma",

email: "mohit@gmail.com"

},

{

name: "Rohan Verma",

email: "rohan@gmail.com"

},

{

name: "Shivam Verma",

email: "shivam@gmail.com"

}

]

}

}

}

</script>

HTML Binding

<template>

<div class="container">

<h1>HTML Binding</h1>

{{ name }}

<div v-html="tag"></div>

</div>

</template>

<script>

export default {

name: "HtmlBinding",

data() {

return {

name: "Mohit Verma",

tag: "<h3>Web Designer</h3>"

}

}

}

</script>

Class Binding

<template>

<div class="container">

<h1>Class Binding</h1>

<div :class="{ bgColor: setColor }">

<h3>Mohit</h3>

</div>

<button @click="setColor = !setColor">Toggle Class</button>

</div>

</template>

<script>

export default {

name: "CssBinding",

data() {

return {

setColor: true

}

}

}

</script>

<style scoped>

.bgColor {

background-color: yellow;

}

</style>

Multiple Classes binding

<template>

<div class="container">

<h1>Multiple Class Binding</h1>

<div :class="setClasses">

<h3>Mohit</h3>

</div>

<button @click="toggleClasses()">Toggle Multiple Class</button>

</div>

</template>

<script>

export default {

name: "CssBinding",

data() {

return {

setBgColor: true,

setFontColor: true

}

},

computed: {

setClasses() {

return {

bgColor: this.setBgColor,

color: this.setFontColor

}

}

},

methods: {

toggleClasses() {

this.setBgColor = !this.setBgColor;

this.setFontColor = !this.setFontColor;

}

}

}

</script>

<style scoped>

.bgColor {

background-color: yellow;

}

.color {

color: red

}

</style>

Ref in Vue Js

<template>

<section>

<div class="container">

<div class="form\_wrap">

<form>

<div class="mb-3">

<label class="form-label">Name</label>

<input type="text" class="form-control" v-model="name" ref="nameField">

<p>{{ getValues() }}</p>

</div>

<button type="button" class="btn btn-primary" @click="focusEnable">Focus</button>

</form>

</div>

</div>

</section>

</template>

<script>

export default {

name: "RefVue",

data() {

return {

name: ""

}

},

methods: {

getValues() {

return `Name Value = ${this.name}`;

},

focusEnable() {

console.log(this.$refs.nameField.value);

}

}}

</script>

Simple Form

<template>

<section>

<div class="container">

<div class="form\_wrap">

{{ form }}

<form>

<div class="mb-3">

<label for="exampleInputEmail1" class="form-label">Email address</label>

<input type="email" class="form-control" v-model="form.email"

ref="emailField">

</div>

<div class="mb-3">

<label for="exampleInputPassword1" class="form-label">Password</label>

<input type="password" class="form-control" v-model="form.password"

ref="passField">

</div>

<button type="submit" class="btn btn-primary" @click="getValues">Submit</button>

</form>

</div>

</div>

</section>

</template>

<script>

export default {

name: "SimpleForm",

data() {

return {

form: {

email: "",

password: "",

}

}

},

methods: {

getValues(*event*) {

*event*.preventDefault();

console.log(`email : ${this.form.email}`);

console.log(`password : ${this.form.password}`);

*// get value by target field using ref*

console.log(`emailRef : ${this.$refs.emailField.value}`);

console.log(`passwordRef : ${this.$refs.passField.value}`);

}

}

}

</script>

Multiple fields form

<template>

<section>

<div class="container">

<div class="form\_wrap">

<form>

<div class="mb-3">

<label class="form-label">Name</label>

<input type="text" class="form-control" v-model="form.name" ref="nameField">

</div>

<div class="mb-3">

<label class="form-label">Email address</label>

<input type="email" class="form-control" v-model="form.email" ref="emailField">

</div>

<div class="mb-3">

<label class="form-label">Password</label>

<input type="password" class="form-control" v-model="form.password" ref="passField">

</div>

<div class="mb-3">

<label class="form-label">Country</label>

<select class="form-select" name="country" ref="countryField" v-model="form.country">

<option selected value="">Select Country</option>

<option value="India">India</option>

<option value="Australia">Australia</option>

<option value="China">China</option>

<option value="Russia">Russia</option>

</select>

</div>

<div class="checks mb-3">

<h3 class="form-label">Skills</h3>

<div class="form-check">

<label class="form-check-label">

<input type="checkbox" class="form-check-input" v-model="form.skills" value="HTML">

HTML

</label>

</div>

<div class="form-check">

<label class="form-check-label">

<input class="form-check-input" type="checkbox" v-model="form.skills" value="CSS">

CSS

</label>

</div>

<div class="form-check">

<label class="form-check-label">

<input class="form-check-input" type="checkbox" v-model="form.skills" value="JS">

JS

</label>

</div>

</div>

<div class="checks mb-3">

<h3 class="form-label">Gender</h3>

<div class="form-check">

<label class="form-check-label">

<input class="form-check-input" type="radio" name="gender" v-model="form.gender"

value="Male">

Male

</label>

</div>

<div class="form-check">

<label class="form-check-label">

<input class="form-check-input" type="radio" name="gender" v-model="form.gender"

value="Female">

Female

</label>

</div>

</div>

<button type="submit" class="btn btn-primary" @click="getValues">Submit</button>

<div class="display\_form\_data">

<pre>{{ form }}</pre>

</div>

</form>

</div>

</div>

</section>

</template>

<script>

export default {

name: "SimpleForm",

data() {

return {

form: {

name: "",

email: "",

password: "",

country: "",

skills: [],

gender: "",

}

}

},

methods: {

getValues(*event*) {

*event*.preventDefault();

}

}

}

</script>

<style>

.display\_form\_data {

position: relative;

padding: 20px;

background: white;

margin-top: 20px;

font-size: 16px;

color: #2196F3;

box-shadow: rgba(0, 0, 0, 0.16) 0px 1px 4px;

}

</style>

Multiple fields form 2 with empty validation

Simple but Lengthy way 👇

<template>

<section>

<div class="container">

<div class="form\_wrap">

<form>

<div class="mb-3">

<label class="form-label">Name</label>

<input type="text" class="form-control" v-model="form.name" ref="nameField">

</div>

<div class="mb-3">

<label class="form-label">Email address</label>

<input type="email" class="form-control" v-model="form.email" ref="emailField">

</div>

<div class="mb-3">

<label class="form-label">Password</label>

<input type="password" class="form-control" v-model="form.password" ref="passField">

</div>

<div class="mb-3">

<label class="form-label">Country</label>

<select class="form-select" name="country" ref="countryField" v-model="form.country">

<option selected value="">Select Country</option>

<option value="India">India</option>

<option value="Australia">Australia</option>

<option value="China">China</option>

<option value="Russia">Russia</option>

</select>

</div>

<div class="checks mb-3">

<h3 class="form-label">Skills</h3>

<div class="form-check">

<label class="form-check-label">

<input type="checkbox" class="form-check-input" v-model="form.skills" value="HTML">

HTML

</label>

</div>

<div class="form-check">

<label class="form-check-label">

<input class="form-check-input" type="checkbox" v-model="form.skills" value="CSS">

CSS

</label>

</div>

<div class="form-check">

<label class="form-check-label">

<input class="form-check-input" type="checkbox" v-model="form.skills" value="JS">

JS

</label>

</div>

</div>

<div class="checks mb-3">

<h3 class="form-label">Gender</h3>

<div class="form-check">

<label class="form-check-label">

<input class="form-check-input" type="radio" name="gender" v-model="form.gender"

value="Male">

Male

</label>

</div>

<div class="form-check">

<label class="form-check-label">

<input class="form-check-input" type="radio" name="gender" v-model="form.gender"

value="Female">

Female

</label>

</div>

</div>

<button type="submit" class="btn btn-primary" @click="getValues">Submit</button>

<div class="display\_form\_data" :class="{ show: setShow }">

<pre>{{ form }}</pre>

</div>

</form>

</div>

</div>

</section>

</template>

<script>

export default {

name: "SimpleForm",

data() {

return {

form: {

name: "",

email: "",

password: "",

country: "",

skills: [],

gender: "",

},

setShow: false

}

},

methods: {

getValues(*event*) {

*event*.preventDefault();

this.setShow = false;

let nameVal, emailVal, passwordVal, countryVal, skillsVal, genderVal;

nameVal = this.form.name;

emailVal = this.form.email;

passwordVal = this.form.password;

countryVal = this.form.country;

skillsVal = this.form.skills;

genderVal = this.form.gender;

if ((nameVal != "") && (emailVal != "") && (passwordVal != "") && (countryVal != "") && (skillsVal != "") && (genderVal != "")) {

this.setShow = true;

} else {

alert("Please enter values correctly before submit");

}

}

}

}

</script>

<style>

.display\_form\_data {

position: relative;

padding: 20px;

background: white;

margin-top: 20px;

font-size: 16px;

color: #2196F3;

box-shadow: rgba(0, 0, 0, 0.16) 0px 1px 4px;

display: none;

}

.display\_form\_data.show {

display: block;

}

</style>

Multiple fields form 2 with empty validation

Short way using loop 👇

<template>

<section>

<div class="container">

<div class="form\_wrap">

<form>

<div class="mb-3">

<label class="form-label">Name</label>

<input type="text" class="form-control" v-model="form.name" ref="nameField">

</div>

<div class="mb-3">

<label class="form-label">Email address</label>

<input type="email" class="form-control" v-model="form.email" ref="emailField">

</div>

<div class="mb-3">

<label class="form-label">Password</label>

<input type="password" class="form-control" v-model="form.password" ref="passField">

</div>

<div class="mb-3">

<label class="form-label">Country</label>

<select class="form-select" name="country" ref="countryField" v-model="form.country">

<option selected value="">Select Country</option>

<option value="India">India</option>

<option value="Australia">Australia</option>

<option value="China">China</option>

<option value="Russia">Russia</option>

</select>

</div>

<div class="checks mb-3">

<h3 class="form-label">Skills</h3>

<div class="form-check">

<label class="form-check-label">

<input type="checkbox" class="form-check-input" v-model="form.skills" value="HTML">

HTML

</label>

</div>

<div class="form-check">

<label class="form-check-label">

<input class="form-check-input" type="checkbox" v-model="form.skills" value="CSS">

CSS

</label>

</div>

<div class="form-check">

<label class="form-check-label">

<input class="form-check-input" type="checkbox" v-model="form.skills" value="JS">

JS

</label>

</div>

</div>

<div class="checks mb-3">

<h3 class="form-label">Gender</h3>

<div class="form-check">

<label class="form-check-label">

<input class="form-check-input" type="radio" name="gender" v-model="form.gender"

value="Male">

Male

</label>

</div>

<div class="form-check">

<label class="form-check-label">

<input class="form-check-input" type="radio" name="gender" v-model="form.gender"

value="Female">

Female

</label>

</div>

</div>

<button type="submit" class="btn btn-primary" @click="getValues">Submit</button>

<div class="display\_form\_data" :class="{ show: setSuccess }">

<pre>{{ form }}</pre>

</div>

<div class="display\_form\_error" :class="{ show: setError }">

<pre>{{ error }}</pre>

</div>

</form>

</div>

</div>

</section>

</template>

<script>

export default {

name: "SimpleForm",

data() {

return {

form: {

name: "",

email: "",

password: "",

country: "",

skills: [],

gender: "",

},

error: [],

setSuccess: false,

setError: false,

}

},

methods: {

getValues(*event*) {

*event*.preventDefault();

this.error = [];

for (const item in this.form) {

if (this.form[item] == "") {

this.error.push(item);

}

}

if (this.error.length == 0) {

this.setSuccess = true;

this.setError = false;

} else {

this.setSuccess = false;

this.setError = true;

}

console.log(this.error);

}

}

}

</script>

<style>

.display\_form\_data {

position: relative;

padding: 20px;

margin-top: 20px;

font-size: 16px;

color: #0f5132;

background-color: #d1e7dd;

border-color: #badbcc;

box-shadow: rgba(0, 0, 0, 0.16) 0px 1px 4px;

display: none;

}

.display\_form\_data.show {

display: block;

}

.display\_form\_error {

position: relative;

padding: 20px;

margin-top: 20px;

font-size: 16px;

color: #842029;

background-color: #f8d7da;

border-color: #f5c2c7;

box-shadow: rgba(0, 0, 0, 0.16) 0px 1px 4px;

display: none;

}

.display\_form\_error.show {

display: block;

}

</style>

Form Modifiers

### **.lazy**

By default, v-model syncs the input with the data after each input event (with the exception of IME composition as [stated above](https://vuejs.org/guide/essentials/forms#vmodel-ime-tip)). You can add the lazy modifier to instead sync after change events:

templ<input v-model.lazy="msg" />

### **.number**

If you want user input to be automatically typecast as a number, you can add the number modifier to your v-model managed inputs:

t<input v-model.number="age" />f the value cannot be parsed wi

The number modifier is applied automatically if the input has type="number".

### **.trim**

If you want whitespace from user input to be trimmed automatically, you can add the trim modifier to your v-model-managed inputs:

template

<input v-model.trim="msg" />

## Use All modifier 👇

<template>

<section>

<div class="container">

<div class="form\_wrap">

<form>

<div class="mb-3">

<label class="form-label">Name</label>

<input type="text" class="form-control" v-model.lazy.number.trim="form.name" ref="nameField">

</div>

<button type="submit" class="btn btn-primary">Submit</button>

<div class="display\_form\_data show">

<pre>Name = {{ form.name }}</pre>

<pre>Type = {{ typeof (form.name) }}</pre>

</div>

</form>

</div>

</div>

</section>

</template>

<script>

export default {

name: "FormModifiers",

data() {

return {

form: {

name: "",

}

}

}

}

</script>

Computed in Vue Js

Computed properties should be used for deriving new data based on existing data, not for performing side effects or modifying data directly.

You should use a method instead of a computed property for the incFun functionality because it is meant to perform an action (incrementing the count), not compute a value based on other properties.

<template>

<div>

<h1>Table Of {{ tableOf }} \* 10 </h1>

<div>{{ getTable }}</div>

</div>

</template>

<script>

export default {

name: "ComputedProp",

data() {

return {

tableOf: 2,

tableLength: 10,

}

},

*// computed is faster than functions defined in methods (computed works on cache based)*

computed: {

getTable() {

return this.tableOf \* this.tableLength;

}

}

*// normally functions defined in methods*

*// methods: {*

*// getTable() {*

*// return this.tableOf \* this.tableLength;*

*// }*

*// }*}

</script>

Watchers in Vue Js

**1**.) In Vue.js, the **watch** option allows you to **watch** for changes to a specific data property and execute a callback function when that property changes. This is particularly useful for performing side effects or computations based on changes to data properties.  
**2**.) Inside the watch object, we define a property named **count**, which corresponds to the **count data property** we want to watch.

**3**.) The value of the count property is a function that takes two arguments: **newValue** and **oldValue**. These represent the new and old values of the count property, respectively.

Here's how you can use the **watch** option in a Vue component: 👇

<template>

<div>

<h1>WatchersVue</h1>

<h3>{{ count }}</h3>

<button @click="incFun()">Inc</button>

<button @click="decFun()">Dec</button>

</div>

</template>

<script>

export default {

name: "WatchersVue",

data() {

return {

count: 0,

}

},

watch: {

count(*val*, *prev*) {

if (*val* > *prev*) {

console.log("Inc");

} else {

console.log("Dec");

}

}

},

methods: {

incFun() {

this.count = this.count + 1;

},

decFun() {

this.count = this.count - 1;

}

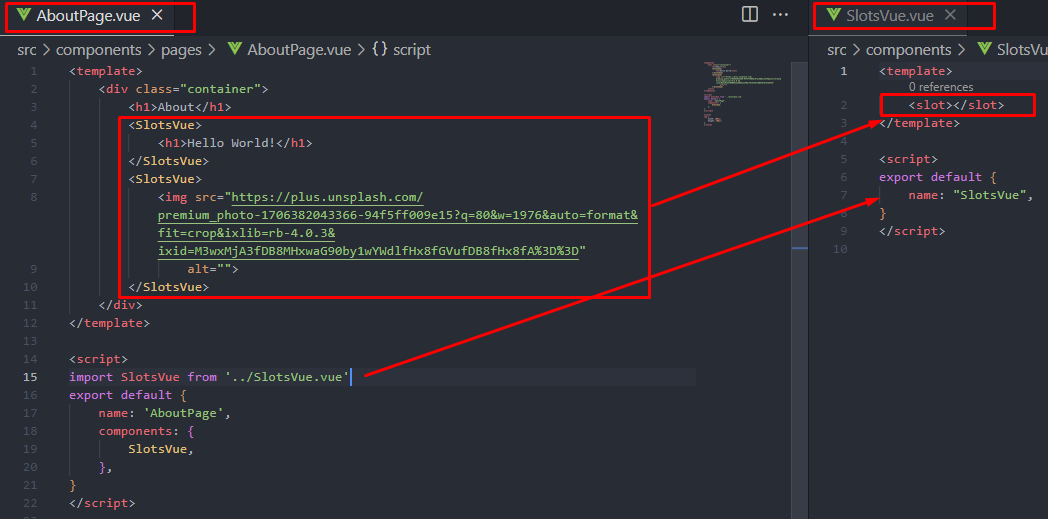
},

}

</script>

Slots in Vue Js

**Yes, in Vue.js, you can pass template content to a child component using what is known 👇**



**App.vue** 👇

<template>

<div class="container">

<h1>About</h1>

<SlotsVue>

<h1>Hello World!</h1>

</SlotsVue>

<SlotsVue>

<img src="https://plus.unsplash.com/premium\_photo-1706382043366-94f5ff009e15?q=80&w=1976&auto=format&fit=crop&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D"

alt="">

</SlotsVue>

</div>

</template>

<script>

import SlotsVue from '../SlotsVue.vue'

export default {

name: 'AboutPage',

components: {

SlotsVue,

},

}

</script>

**SlotsVue.vue** 👇

<template>

<slot></slot>

</template>

<script>

export default {

name: "SlotsVue",

}

</script>

Named **slots** allow you to define **placeholders** within the **child component's** template where the **parent component** can insert specific content. This gives the **parent component** control over the structure and layout of the **child component's** template.

Here's a basic example of how you can use named slots:

**ParentComponent**.vue:👇

<template>

<div>

<ChildComponent>

<template v-slot:header>

<h2>This is the header</h2>

</template>

<template v-slot:content>

<p>This is the content</p>

</template>

</ChildComponent>

</div>

</template>

<script>

import ChildComponent from './ChildComponent.vue';

export default {

components: {

ChildComponent

}

}

</script>

**ChildComponent**.vue:👇

<template>

<div>

<div class="header">

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

</div>

<div class="content">

<slot name="content"></slot>

</div>

</div>

</template>

Send data from router to component

**Header file 👇**

<router-link :to="{path: '/product-message', query: {id: 'marketingOtp'}}">

<div class="text\_wrap tabClickLink" data-tabclick="marketingOtp" data-url="product-message">

<h4>

<img src="../assets/images/hdr\_icon\_02.svg" class="tm\_hdr\_icon" />

Marketing SMS

</h4>

<p>

Promotion of brands and <br />

products

</p>

</div>

</router-link>

**Footer file 👇**

**<script>**

**export default {**

**name: "FooterEng",**

**data() {**

**return {**

**changeTabFromUrl: "",**

**}**

**},**

**mounted() {**

***// Pehli baar ID ko retrieve karna***

**this.getIdfromUrl();**

***// Vue Router ka afterEach hook register karna***

**this.$router.afterEach(() => {**

***// Har inner page change hone par ID ko retrieve karna***

**this.getIdfromUrl();**

**});**

**},**

**methods: {**

***// changeTab function***

**changeTab() {**

**console.log("changeTabFromUrl " + this.changeTabFromUrl);**

**const tabBtn = `[data-bs-target="#${this.changeTabFromUrl}"]`;**

**console.log(tabBtn);**

**setTimeout(() => {**

**const targetElement = document.querySelector(tabBtn);**

**if (targetElement) {**

**targetElement.click();**

**document.querySelector(".prod\_desc\_section").scrollIntoView({**

**behavior: 'smooth'**

**});**

**} else {**

**console.error(`Element with selector ${tabBtn} not found.`);**

**}**

**}, 1000);**

**},**

***// getIdfromUrl for active tab on change***

**getIdfromUrl() {**

***// Pura URL le lena ke bajayf sirf query string le lena.***

**const queryString = window.location.search;**

**const urlParams = new URLSearchParams(queryString);**

***// ID parameter ko extract karna.***

**const id = urlParams.get('id');**

**if (id) {**

**this.changeTabFromUrl = id;**

**this.changeTab();**

**}**

**},**

**}**

**}**

**</script>**

Language Switcher

**HeaderGlobal.vue file 👇**

**<template>**

**<HeaderEng :languageChangeFun="languageChange" v-if="isLangEng" />**

**<HeaderCh :languageChangeFun="languageChange" v-if="!isLangEng" />**

**<router-view></router-view>**

**<FooterEng :languageChangeFun="languageChange" v-if="isLangEng" />**

**<FooterCh :languageChangeFun="languageChange" v-if="!isLangEng" />**

**</template>**

**<script>**

**import HeaderEng from './HeaderEng.vue';**

**import HeaderCh from './HeaderCh.vue';**

**import FooterEng from './FooterEng.vue';**

**import FooterCh from './FooterCh.vue';**

**export default {**

**name: "HeaderGlobal",**

**components: {**

**HeaderEng,**

**HeaderCh,**

**FooterEng,**

**FooterCh**

**},**

**data() {**

**return {**

**isLangEng: true**

**}**

**},**

**methods: {**

**languageChange(*language*) {**

**console.log(*language*);**

**if (*language* === "EN") {**

**this.isLangEng = true;**

**} else {**

**this.isLangEng = false;**

**}**

**this.changePageAccordingLang();**

**},**

**changePageAccordingLang() {**

**let currPath = this.$route.path;**

**let currPathWithoutSlash = currPath.substring(1);**

**let dynRoot = "";**

***// Check if the current route has a path***

**if ((currPathWithoutSlash) && (currPathWithoutSlash != "zh")) {**

**if ((this.isLangEng == true)) {**

**dynRoot = currPathWithoutSlash.substring(3);**

**this.$router.push({**

**path: "/" + dynRoot**

**});**

***// Redirect to English version route***

**} else {**

***// Redirect to Chinese version route***

**dynRoot = "/zh-" + currPathWithoutSlash;**

**this.$router.push({**

**path: dynRoot**

**});**

**}**

**} else {**

**if (this.isLangEng == true) {**

***// Redirect to English version route***

**this.$router.push({**

**path: '/'**

**});**

**} else {**

***// Redirect to Chinese version route***

**this.$router.push({**

**path: '/zh'**

**});**

**}**

**}**

**}**

**}**

**}**

**</script>**

**HeaderCh.vue file 👇**

<template>

*<!-- header start -->*

<div class="right">

<router-link to="/login" class="text\_btn">Login</router-link>

<router-link to="/sign-up" class="primary\_btn">Get Started</router-link>

<div class="lang">

<select name="lang" id="lang" @change="languageChangeFun(language)" v-model="language">

<option value="EN" selected>EN</option>

<option value="CN" >中文</option>

</select>

</div>

<button class="toggle-menu"></button>

</div>

</header>

</template>

<script>

export default {

name: 'HeaderEng',

data() {

return {

language: "EN"

}

},

props: {

languageChangeFun: Function

},

watch: {

language(*val*, *prev*) {

console.log(*prev*);

sessionStorage.setItem("language", *val*);

}

},

methods: {

*// Function to update component data with session data*

updateDataFromSession() {

*// Get session data*

const language = sessionStorage.getItem("language");

*// Parse session data if available*

if (language) {

console.log("session" + language);

this.language = language;

this.languageChangeFun(language);

}

}

},

created() {

*// Call the updateDataFromSession method when the component is created*

this.updateDataFromSession();

}

}

</script>

MarkDown in Vue Js

npm install markdown-it

npm install raw-loader --save-dev

<template>

<div class="developer\_page">

<ul class="sidebar\_options\_wrap">

<li>

<a href="javascript:void(0);" target="\_blank" data-github="developer-pages-api-en/message/mt-sending">

Send Email Verification Code

</a>

</li>

</ul>

</div>

</template>

<script>

import MarkdownIt from "markdown-it";

const md = new MarkdownIt();

export default {

name: 'DeveloperPage',

data() {

return {

markdownContent: "",

renderedMarkdown: "",

currBtnData: "" *// Initialize currBtnData here*

};

},

methods: {

async getDataApi(*url*) {

try {

console.log("url: ", *url*);

*// Assuming url includes the correct path*

const markdownFile = await import(`!!raw-loader!/${*url*}.md`);

this.markdownContent = markdownFile.default;

this.renderedMarkdown = md.render(this.markdownContent);

} catch (error) {

console.error("Error loading Markdown file:", error);

}

},

handleClick(*event*) {

*event*.preventDefault();

const currBtn = *event*.currentTarget;

const currGitData = currBtn.getAttribute("data-github");

this.currBtnData = currGitData;

console.log("currBtnData: ", this.currBtnData);

this.getDataApi(this.currBtnData);

}

},

mounted() {

const sidebarOptions = document.querySelectorAll(".developer\_page .sidebar\_options\_wrap a");

sidebarOptions.forEach(*option* => {

*option*.addEventListener('click', this.handleClick);

});

}

};

</script>

MarkDown editor online

Online Editors for MarkDown files

<https://stackedit.io/app>

Create flow charts for MarkDown files

<https://mermaid.js.org/intro/>

Header Toggle with jQuery in Vue Js

**HeaderEng.vue**👇

methods: {

addSubmenuIconClick() {

setTimeout(() => {

$("header .submenu\_arrow").click(function () {

$(this).toggleClass("active");

$(this).closest(".has\_submenu").find(".submenu").slideToggle();

});

}, 1000);

},

headerToggle() {

$(".toggle-menu").toggleClass("active");

$("header .header\_inner\_wrap .inner .navigation\_wrap").slideToggle();

},

hideHeader() {

if ($(window).width() < 1200) {

*// Execute the code snippet only when the screen width is less than 1200 pixels*

if ($("header .toggle-menu").hasClass("active")) {

$("header .toggle-menu").removeClass("active");

}

$("header .header\_inner\_wrap .inner .navigation\_wrap").slideUp();

if ($("header .submenu\_arrow").hasClass("active")) {

$("header .submenu\_arrow").removeClass("active");

}

$("header .has\_submenu .submenu").slideUp();

}

}

},

created() {

this.addSubmenuIconClick();

},

mounted() {

this.hideHeader();

this.$router.afterEach(() => {

this.hideHeader();

});

},

Locations dynamically add from Obj in Vue Js

**CompanyPage.html 👇**

<ul class="locations">

<li v-for="(location, index) in locations" :key="index" class="location" @mouseover="showLocation(index)" @mouseleave="hideLocation(index)" :class="[location.countryClass, { 'active': activeLocation === index }]">

<div class="lcmark">

<div class="icon">

<img src="../../assets/images/location\_mark.png" alt="location\_mark" />

</div>

<h4 class="country">{{ location.country }}</h4>

</div>

<div class="info">

<h5>{{ location.country }}</h5>

<a v-if="location.contactNumber" :href="'tel:' + location.contactNumber" class="tel"><span class="strong">Tel:</span> {{ location.contactNumber }}</a>

<p v-if="location.address">

<span class="strong">Address: </span> {{ location.address }}

</p>

</div>

</li>

</ul>

<script>

export default {

name: 'CompanyPage',

data() {

return {

activeLocation: null,

locations: [{

country: "Shenzhen, China",

countryClass: "shenzhen",

contactNumber: "400-7800-900",

address: "6/F Room601 Building A1, Kexing Science Park No .15 Keyuan Road, Nanshan District Shenzhen 518000, Guangdong Province China ",

},

]

}

},

methods: {

showLocation(*index*) {

this.activeLocation = *index*; *// Set active location index when hovering*

},

hideLocation() {

this.activeLocation = null; *// Reset active location index when mouse leaves*

}

}

}

</script>

History dynamically add from Obj in Vue Js

*<!-- History -->*

<section class="history\_sec pb\_200" data-aos="fade-up" data-aos-delay="0">

<div class="container">

<div class="sec\_heading text-center">

<h2>A brief history of NXCLOUD</h2>

</div>

<div class="content">

<div class="history\_wrapper">

<div class="inner">

<ul class="history\_listing">

<li :class="{ 'active': activeHistory === index }" v-for="(history, index) in histories" :key="index" @mouseover="showHistory(index)" >

<div class="left">

<button data-history="history\_1">

<span>{{history.year}}</span>

</button>

</div>

<div class="right">

<div class="text\_wrap" v-for="item in history.content" :key="item">

<h3>{{item.title}}</h3>

<p>{{item.desc}}</p>

</div>

</div>

</li>

</ul>

</div>

</div>

</div>

</div>

</section>

<script>

export default {

name: 'CompanyPage',

data() {

return {

activeHistory: 0,

histories: [{

year: 2018,

content: [{

title: "March",

desc: "Official founding of NXCLOUD in Singapore, with branches and offices in the USA, Indonesia, and Malaysia."

},

{

title: "August",

desc: "Official launch of Independent R&D cloud communication platform."

},

]

}

{

year: 2021,

content: [{

title: "May",

desc: "NXCLOUD joined the GSMA Association and became its official corporate member."

}, ]

},

],

}

},

methods: {

showLocation(index) {

this.activeLocation = index; *// Set active location index when hovering*

},

hideLocation() {

this.activeLocation = null; *// Reset active location index when mouse leaves*

},

showHistory(index) {

this.activeHistory = index; *// Set active History index when hovering*

},

}

}

</script>

Developer page search functionality in Vue Js

<template>

<div *class*="search\_wrap">

<form>

<input *type*="text" *class*="search\_field" *placeholder*="Document Keyword search..." *v-model*="docSearchVal" @input="docSearchFun()" />

<button *class*="search\_btn"></button>

</form>

<div *v-if*="resultNotFound" *class*="resultNotFound">

<p *class*="alert alert-danger" *role*="alert">Sorry, we couldn't find any results</p>

</div>

</div>

</template>

<script>

export default {

name: 'DeveloperPage',

data() {

return {

docSearchVal: "",

resultNotFound: false,

};

},

methods: {

docSearchFun() {

const sidebar\_options\_wrap = document.querySelector(".sidebar\_options\_wrap");

const sidebar\_accordion\_items = sidebar\_options\_wrap.querySelectorAll(".accordion-item");

const sidebar\_accordion\_ul = sidebar\_options\_wrap.querySelectorAll("ul");

const sidebar\_all\_options = sidebar\_options\_wrap.querySelectorAll("li");

let docSearchVal = this.docSearchVal;

docSearchVal = docSearchVal.toLowerCase();

let optionTextNode, optionText, anchor;

sidebar\_all\_options.forEach(*option* => {

if (*option*.querySelector(".text b")) {

optionText = *option*.querySelector(".text b").textContent.toLowerCase();

optionTextNode = optionText;

} else {

anchor = *option*.querySelector("a").textContent.toLowerCase();

optionTextNode = anchor;

}

*// check list item's inner text with input's value*

if ((optionTextNode.indexOf(docSearchVal) != -1)) {

*option*.classList.add("show");

*option*.classList.remove("hide");

} else {

*option*.classList.add("hide");

*option*.classList.remove("show");

}

});

*// hide sidebar\_accordion\_item if all options are hide*

sidebar\_accordion\_ul.forEach(*ul* => {

const liAreFound = *ul*.querySelectorAll("li.show");

if (!liAreFound.length) {

*ul*.closest(".accordion-item").classList.add("hide");

*ul*.closest(".accordion-item").classList.remove("show");

} else {

*ul*.closest(".accordion-item").classList.add("show");

*ul*.closest(".accordion-item").classList.remove("hide");

}

});

*// show error if search not found*

let isAnyHideFields = true;

sidebar\_accordion\_items.forEach(*item* => {

if (*item*.classList.contains("show")) {

isAnyHideFields = false;

return;

}

});

this.resultNotFound = isAnyHideFields;

},

}

};

</script>

\*created Lifecycle Hook:\*

**\*created Lifecycle Hook:\***

The created hook is called after the Vue instance has been created but before it is mounted to the DOM.

You can perform actions such as initializing data, setting up event listeners, or making API requests in this hook.

It is commonly used for tasks that need to be executed once the component instance is created.

**\*beforeMount Lifecycle Hook:\***

The beforeMount hook is called right before the Vue instance is mounted to the DOM.

It is useful for tasks that need to be performed just before the rendering process begins.

At this point, the template has been compiled, and the virtual DOM is about to be rendered.

**\*mounted Lifecycle Hook:\***

The mounted hook is called after the Vue instance has been mounted to the DOM.

It is commonly used for actions that require access to the DOM, such as setting up third-party libraries, fetching additional data, or interacting with the DOM directly.

This is often the place to perform asynchronous tasks that depend on the component being in the DOM

**\*watch Option:\***

The watch option allows you to watch for changes in data properties and execute a function when the specified property changes.

It is useful for reacting to changes in data and performing actions in response to those changes.

**CRATED and beforemounted**

While both hooks occur during the initialization phase, created is more focused on general initialization tasks, while beforeMount is specifically aimed at tasks to be performed just before the component is mounted to the DOM. The key distinction is the timing of when the DOM updates occur in relation to the execution of these hooks.

In most cases, you might find yourself using the created hook for general setup and the beforeMount hook for any tasks that should be executed immediately before the component is rendered to the DOM. However, the specific use cases can vary based on the requirements of your application.

**watch**

In Vue.js, the watch option is used to observe changes in data properties and take specific actions when those properties change. It allows you to react to changes in data and perform tasks based on those changes. The watch option is particularly useful for scenarios where you need to respond to changes in data and trigger additional logic.