**Vue JS**

Installation

VUE Commands

Install

npm install -g @vue/cli

vue create project-name

cd project-name

npm run serve

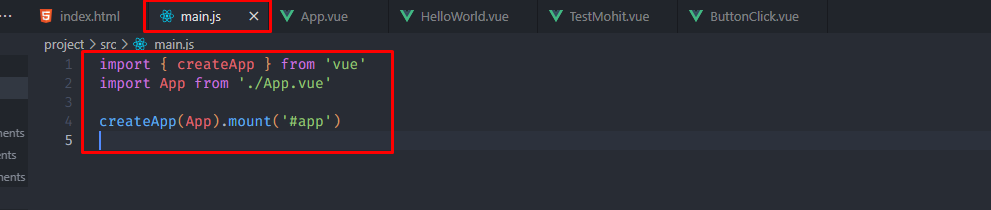
Step for Build

npm run build

npm install -g serve

serve

Main.js

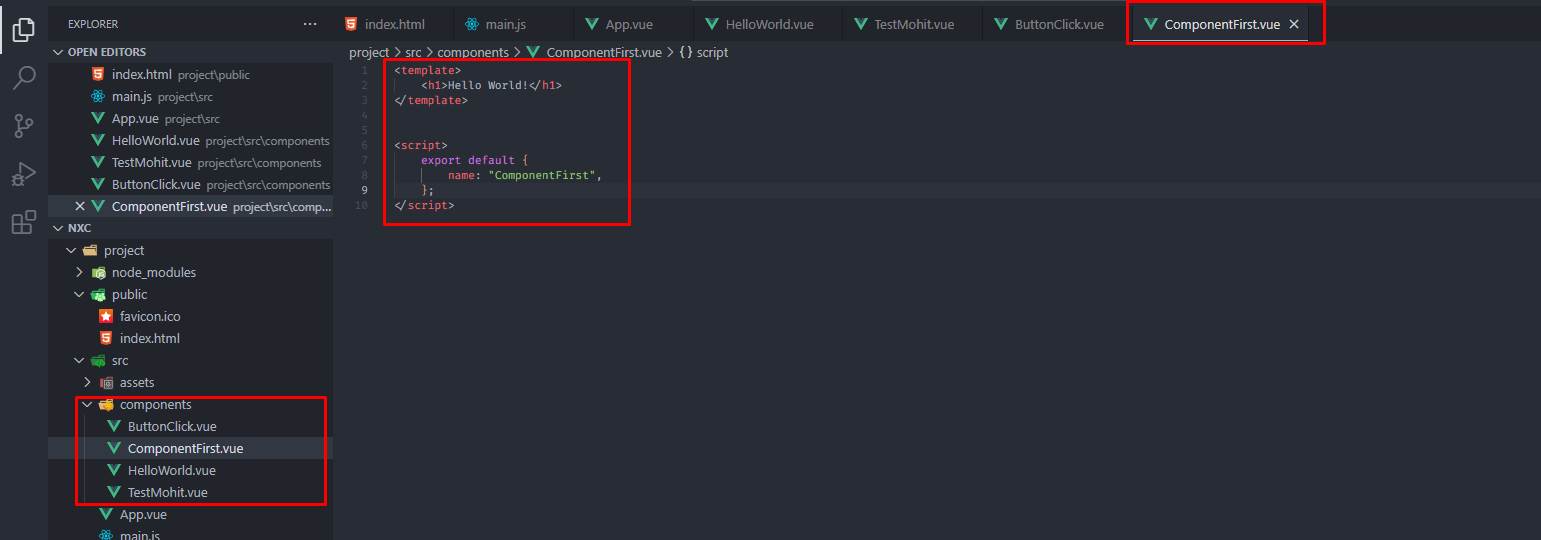


import { createApp } from 'vue'

import App from './App.vue'

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

Create Simple Component



<template>

<h1>Hello World!</h1>

</template>

<script>

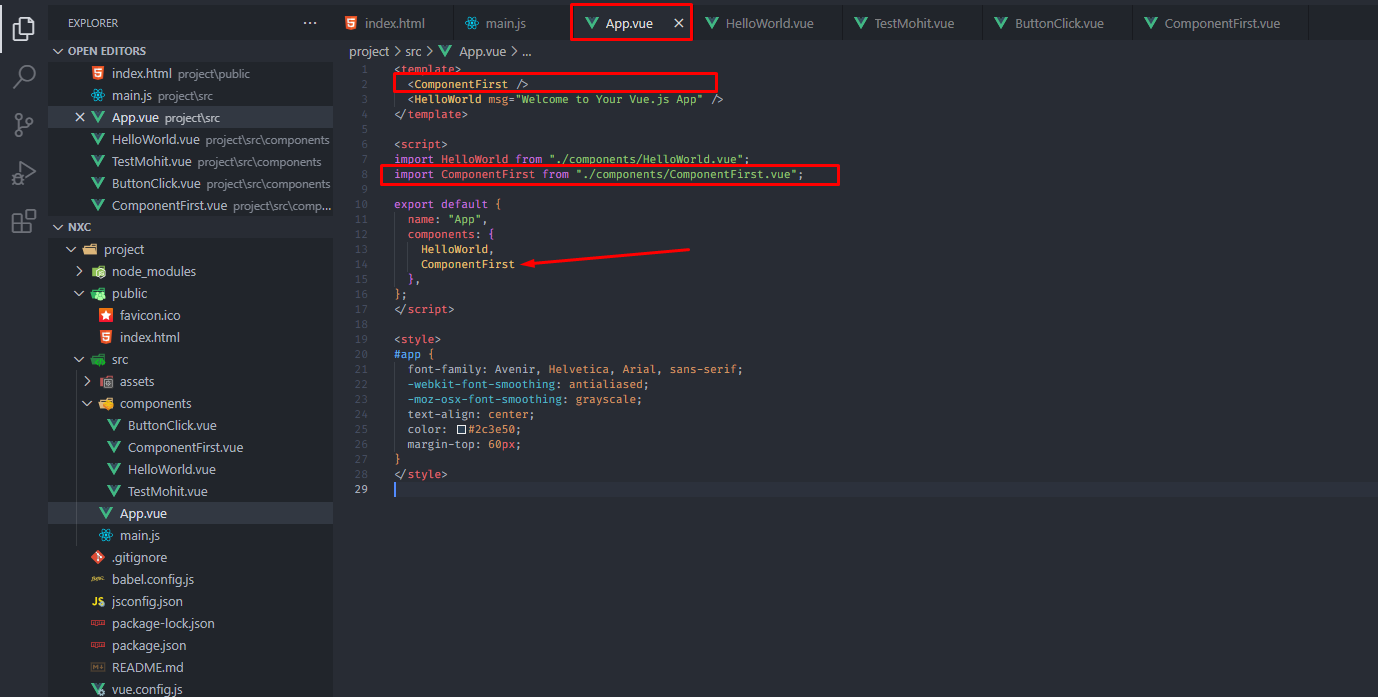
export default {

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;

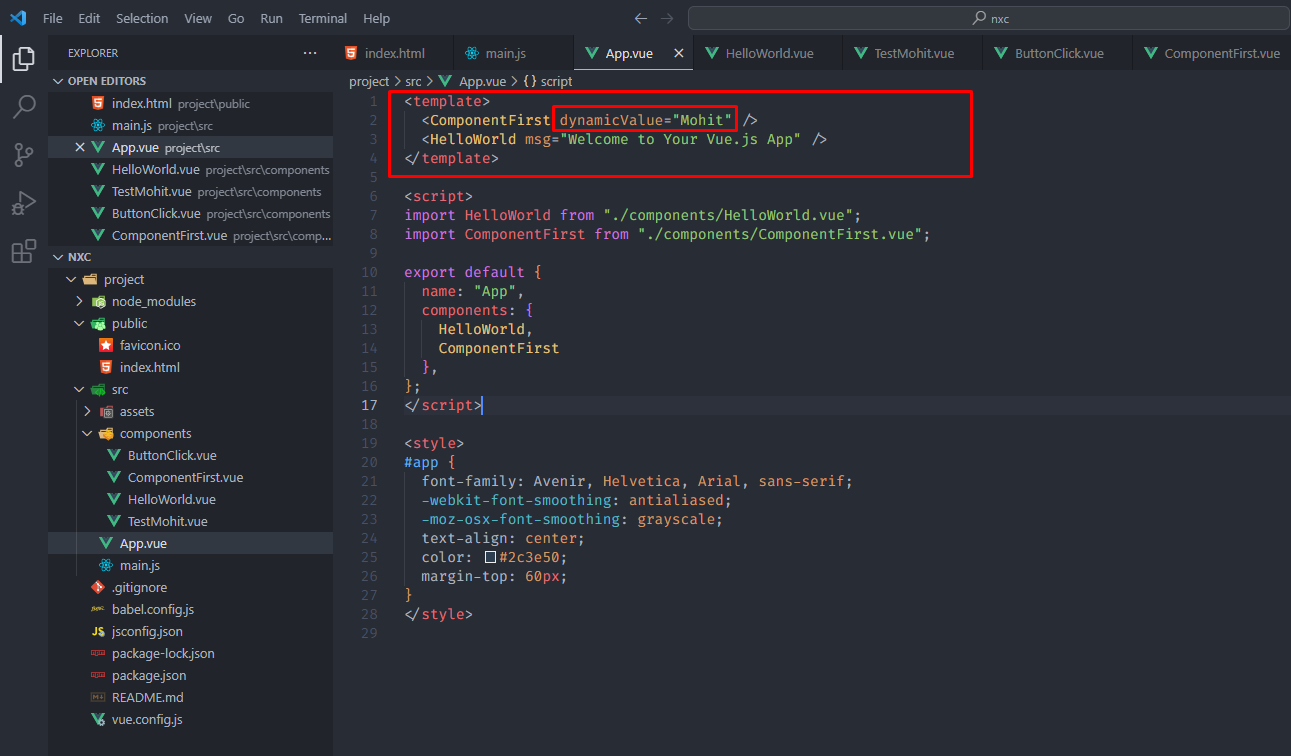
}

</style>

Props

Props for get data from file where comp called

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

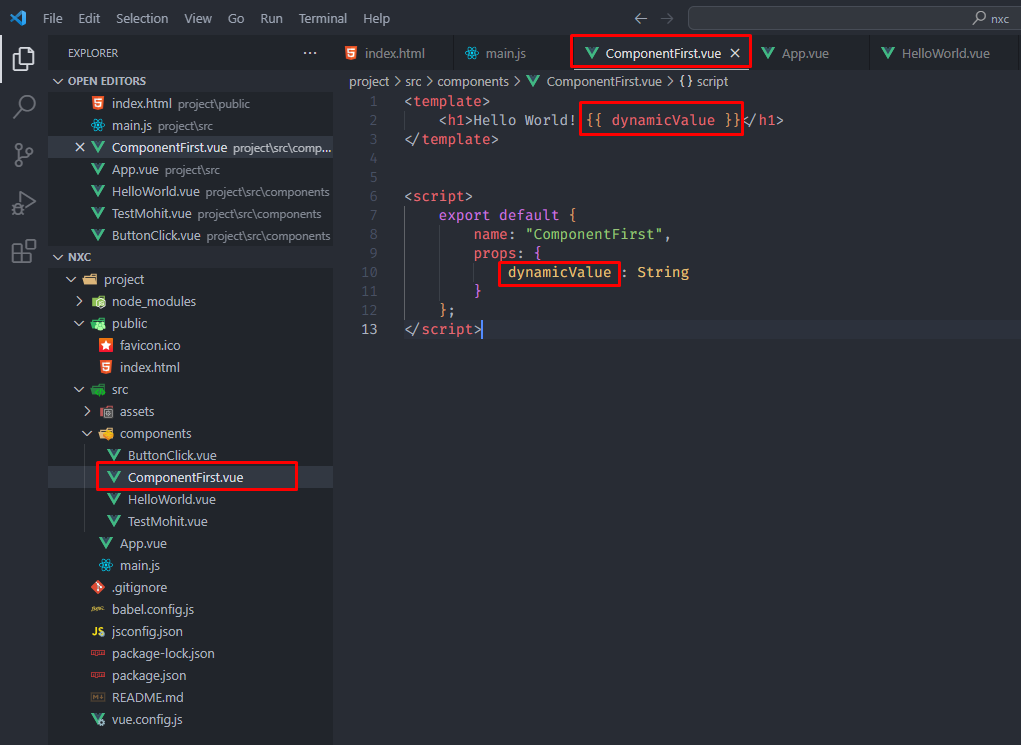
},

};

</script>

Access props in component

.



<template>

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

</template>

<script>

export default {

name: "ComponentFirst",

props: {

dynamicValue : String

}

};

</script>

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>