Vue.Js Setup Part 2 :

**++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++**

**links :** [**https://www.youtube.com/watch?v=YdQArrf3zLY&list=PL8p2I9GklV45qwTH-mdzllUuFRJO-euYn&index=13**](https://www.youtube.com/watch?v=YdQArrf3zLY&list=PL8p2I9GklV45qwTH-mdzllUuFRJO-euYn&index=13)

**>>>> props in vue js >>>**

**what is props >>** this is function i pass the parameter , like wise i pass props in templates.

if i pass the props in value in child components or siblling components.

props pass normally as one component to another component

but if i pass data based on name or userName then use to

**v-bind:name="userName"**

**fetch single name based >>>**

<template>

<div id="app">

<h2>Props in vue Js</h2>

<!-- <Child name="Sauarbh Shukla" /> -->

<Child v-bind:name="userName" />

</div>

</template>

<script>

import Child from './components/Child.vue'

export default {

name: 'App',

components: {

Child,

},

data()

{

return {

userName:"SAM"

}

}

}

</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>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

If i got array stored data then ...

Pass multiple data via parent to child >>>

**App.vue >>>**

<template>

<div id="app">

<h2>Props in Vue.js</h2>

<!-- Bind the first user's name to the Child component -->

<Child v-bind:name="users[0].name" />

<!-- If you want to loop through all users -->

<div v-for="(user, index) in users" :key="index">

<Child v-bind:name="user" />

</div>

</div>

</template>

<script>

import Child from './components/Child.vue';

export default {

name: 'App',

components: {

Child,

},

data() {

return {

users: [

{ name: 'Anil' ,email:"anil@gmail.com" },

{ name: 'Mukesh' , email:"mukesh@gmail.com"},

],

};

},

};

</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>

**Child.vue >>>**

<template>

<div>

<p>{{ name.name }}</p>

<p>{{ name.email }}</p>

</div>

</template>

<script>

export default {

name:"ChildPage",

props: ['name']

};

</script>

**++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++**

**Send Props child to parent >>>>>**

**App.vue >>>**

<template>

<div id="app">

<h2>Send Props Child to Parent in Vue.js</h2>

**<h5>{{title}}</h5>**

**<Child v-on:changeTitle="updateTitle($event)" />**

</div>

</template>

<script>

import Child, {} from './components/Child.vue';

export default {

name: 'App',

components: {

Child,

},

**data() {**

**return {**

**title:"Props Title"**

**};**

**},**

**methods:{**

**updateTitle(title){**

**this.title=title**

**}**

**}**

};

</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>

---------------------------------------------------------------------------------------------------------

**Child.vue >>>>**

<template>

<div>

**<button @click="updateTitle">Update Parent Title</button>**

</div>

</template>

<script>

export default {

name:"ChildPage",

**methods:{**

**updateTitle(){**

**this.$emit('changeTitle',"Parent Updated")**

**}**

**}**

}

</script>

child to send parent data use $emit function pass two parameter function and update value ..

in child components >>>>

in tempaltes >>>

**<button @click="updateTitle">Update Parent Title</button>**

in script >>>

**updateTitle(){**

**this.$emit('changeTitle',"Parent Updated")**

**}**

App.vue >>>>>

in tempates tag >>

<h5>{{title}}</h5>

<Child v-on:changeTitle="updateTitle($event)" />

**in script >>>**

data() {

return {

title:"Props Title"

};

},

methods:{

updateTitle(title){

this.title=title

}

}

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

**life cycle methods (Hooks) >>>>**



**first run this process Digram Created >>>**

**new Vue() ===> (default working as main.js file)**

**|**

**init ( Events & Lifecycle )**

**beforeCreate ----------------------------- |**

**|**

**init ( injections & reactivity )**

**created ----------------------------- |**

**Has ( el ) Option? (el means external element, like main.js)**

**| ------- NO**

**Yes**

**|**

**Has "template" option?**

**| <------ vm.$mount(el) is called**

**---------- | ----------**

**Yes No**

**| |**

**Compile template into Compile el's outerHTML**

**render function \* as template \***

**|**

**beforeMount <-------- |**

**Create vm.$el and replace "el" with it**

**|**

**mounted -----------------------------> Component is now rendered and attached to the DOM**

**|**

**Watching data changes (Reactivity)**

**|**

**beforeUpdate <-------- When data changes**

**|**

**Re-render the virtual DOM**

**|**

**updated -----------------------------> Component is re-rendered with updated data**

**|**

**beforeDestroy -------------------------> The component is about to be destroyed**

**|**

**Destroying watchers, child components, event listeners, etc.**

**|**

**destroyed --------------------------> The component is completely destroyed**

Summary >>>

Key Notes:

**beforeCreate**: The lifecycle starts. No data, methods, or components are available yet.

**created**: Data, methods, and computed properties are available. However, the template is not compiled.

**beforeMount**: Before Vue inserts the compiled template into the DOM.

**mounted**: Vue inserts the template into the DOM and the component is visible.

**beforeUpdate**: If data changes, this hook is called before updating the DOM.

**updated**: Called after the data has been re-rendered and updated.

**beforeDestroy**: Triggered before the component is destroyed.

**destroyed**: The component is fully destroyed and detached from the DOM.

code >>

Child.vue >>

<template>

<div>

<h5>This is child component</h5>

</div>

</template>

<script>

export default {

name:"ChildPage",

methods: {

},

data(){

return {}

},

beforeCreate:function(){

console.log("beforeCreate");

},

beforeMount:function(){

console.log("beforeMount");

},

mounted:function(){

console.log("mounted");

},

created:function(){

console.log("created");

}

};

</script>

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

**Form with checkbox and radio button >>>>**