**Passing data from Parent to Component (Child) with Bind**

Parent:

<friends :nameFromParent="title" />

*//* title*is the value we gonna pass through* nameFromParent

in Child

export default {

        props: ["nameFromParent"],

*//* *To Recive* nameFromParent *put it in props array*

<p>{{nameFromParent}}</p>

*//Now we can Use it*

**Passing data from child to parent**

Child:

<button @click="unfriend(friend.name)">

Delete

</button>

*<!--*friend.name *is the value you passing -->*

methods: {

            unfriend(name) {

                this.$emit('delete', {name});

            }

        } *//*name*is the value you passing, you can pass it in object or string or any type*

Parent:

<friends  @delete="deleteFriend"/>

*<!-- Bind the component tag with the name you typed (delete) -->*

methods:{

      deleteFriend(nameFromChild){   
 alert(nameFromChild);

}  
}  
*//nameFromChild is our Child Value we are just passed*

**Vue onload function**

export default {

  beforeMount(){

    alert("Component has been created!");  
 }  
}

**Routing**

<ul class="navbar-nav">

        <li>

          <router-link class="nav-link" to="/home">Home </router-link>

        </li>

        <li>

          <router-link class="nav-link" :to="goPage">About</router-link>

        </li>

*<!-- Route will accept dinamic value  if we type it like this :to  -->*

      </ul>

**Routing ($route)**

this.$route => Print object contain route details like name , fullPath, param (if set) …..

*// how to set a param*  
{

    path: "/ViewProfile/:user\_id",

    name: "ViewProfile",

    component: ViewProfile

  }

*// user\_id is the param key*

Param is the object that contain the value after the / in the link

this.$route.params => { user\_id: "ViewProfile"}   
this.$route.params.user\_id => "ViewProfile"

**Routing ($router)**

goTo(){

      this.$router.push('about')

    }

// Function: go to about path

**Routing classes**

router-link-active => This exact element

router-link-exact-active => This exact element in this exact URL

**Vue LifeCylcle hooks**

**beforeCreate()** => Runs at the very initialization of your component

**create()** => Vue created your component but not rendered it to DOM

**beforeUpdate()** => allows you to get the new change to any data on your component before it actually gets rendered.

**Updated()** => hook runs after data changes on your component and the DOM re-renders. If you need to access the DOM after a property change, here is probably the safest place to do it.

**beforeMount()** => hook runs right before the initial render happens and after the template or render functions have been compiled. Most likely you’ll never need to use this hook. Remember, it doesn’t get called when doing server-side rendering.

**Mounted()** => you will have full access to the reactive component, templates, and rendered DOM (via. this.$el). Mounted is the most-often used lifecycle hook.

**beforeDestroy()** => is fired right before teardown. Your component will still be fully present and functional. If you need to cleanup events or reactive subscriptions, beforeDestroy would probably be the time to do it.

**Destroyed()** => hook to do any last-minute cleanup or inform a remote server that the component was destroyed

**Making Request With Axios**

1. Install Axios => npm install axios
2. Import it and use it

import axios from "axios";

created() {

      axios

        .get("http://www.json-generator.com/api/json/get/ceRevxmPyq?indent=2")

        .then(jsonTest => {

          window.console.log(jsonTest.data);  
 this.posts = jsonTest.data;  
 });  
 }