[Angular 6 pass data between 2 not related components](https://stackoverflow.com/questions/44414226/angular-4-pass-data-between-2-not-related-components)

Greetings Earthlings… welcome back to codewithsrini.

This lecture will primarily be talking about passing data between two non-related components.

Let’s start with an already existing code sample from one of my lecture that deals with @Output function.

In that we managed to pass data from a child component “Login” to a parent component “app.component”.

But here, we cannot use @Output binding function to pass data. Because the data has to be shared to another component inside app.component. Basically the data sharing will be between siblings.

So let’s warm up our application and start building it, we already have all the necessary ingredients.

Firstly, let’s create a new component called navbar and move all our code from app component to navbar component.

**navbar.component.html**

<nav>

  <ul id="mainNav" class="nav nav-fill">

    <li class="nav-item">

      <a href="#" class="nav-link">Dashboard</a>

    </li>

    <li class="nav-item">

      <a href="#" class="nav-link">Projects</a>

    </li>

    <li class="nav-item">

      <a href="#" class="nav-link">Users</a>

    </li>

    <li \*ngIf="!isUserLogged"class="nav-item">

      <a href="#" class="nav-link">Login</a>

    </li>

<li \*ngIf="isUserLogged"class="nav-item">

      <a href="#" class="nav-link">Logout ({{user.user}})</a>

    </li>

  </ul>

</nav>

**Go to app.component.html**

<app-navbar></app-navbar>

<div class="container">

  <!-- <app-test (change)="somethingChanged($event)"></app-test> -->

<login (change)="onLoginEventFired($event)"></login>

</div>

**Create a service called (login.service.ts)**

Create a field in class called $isLoggedIn.

So.

$isLoggedIn = new EventEmitter<any>();

Now go to top and add EventEmitter to our import statement.

The compiler will stop fuzzing about the error.

Create a method called login

login() {

console.log('Login Service to Application');

}

And we gonna emit that change in login state for now.

import { LoggedInUserEventArgs } from './login.component';

Create a user field with LoggedInUserEventArgs as an interface for it’s type.

user:LoggedInUserEventArgs = {

user:"",

usertype: "",

role: ""

};

And then add all those params to user field inside login method;

login() {

console.log('Login Service to Application');

this.user.user = 'CodeWithSrini';

this.user.usertype = 'internal';

this.user.role = 'admin';

this.$isLoggedIn.emit(this.user);

}

This is ideally be executed only when there is a successful login, but this is just an example to demonstrate our routine.

The emit method raises or publish and event for $isLoggedIn variable which then can be subscribed by another component. So that target component can consume the value emitted. In this case, the user details. (well, after the user has logged in)

Now let’s fix few things in our login component class to invoke our login service method.

Goto **login.component.ts**

constructor(

private service: LoginService

) { }

login() {

// this.click.emit({ user: 'codewithsrini', usertype: 'internal', role: 'admin'});

this.service.login();

}

Back in our browser let’s check if thigs are not breaking up so far.

Brilliant now it looks like our login service is working as expected, we believe that user data is published.

Now time to capture or subscribe to that data from another component called nav bar.

Go to **navbar.component.ts**

constructor( private loginService: LoginService) { }

ngOnInit() {

this.loginService.$isLoggedIn

.subscribe((data) => {

console.log("I got data in navbar", data);

});

}

Let’s check our browser to see if we get that message in navbar.

Now we gonna make use of isUserLoggedIn and User field in navbar class to switch login and logout buttons in navbar.

So

In **navbar.component.ts**

isUserLogged = false;

user: LoggedInUserEventArgs;

constructor( private loginService: LoginService) { }

ngOnInit() {

this.loginService.$isLoggedIn

.subscribe((data) => {

console.log("I got data in navbar", data);

this.isUserLogged = true;

this.user = data;

});

}

Back in the browser, hurray its working as expected.

This is how we pass data from one component to another component which are not related. So the data passing is between siblings and not parent child hierarchy.

Thank You for watching this lecture, if you like it, please give it a like and if you have any suggestions or comments do write them below and finally subscribe to my channel to get latest updates.

Thank you