Parent component: App component

Child component: Login component

Parent to child communication : @Input()

Task:

You have created a component called LoginComponent.

Its title should be provided by the parent component AppComponent

Do this using @Input() decorator

Task:

App component to have a text box/ dropdownlist where the color can be typed/ selected.

The child component background color should change according to the color selected

in the parent

1. In LoginComponent ts file, declare a variable and decorate it with @Input() decorator

@Input() backgroundColor:string=”yellow”;

1. In LoginComponent html, use that variable in ngStylew

<div [ngStyle]=”{‘background-color’:backgroundColor}”>

1. In AppComponent html,

<app-login backgroundColor=”red”></app-login>

Task:

Emit a LoginEvent from child component (LoginComponent) and handle that event in parent component (AppComponent)

In css:

{key1:value1;key2:value2}

In JSON:

{key1:value1,key2:value2}

ngClass

apply the class to a html element conditionally.

<div [ngStyle]="{'background-color': backgroundColor,'color':'red'}">

    <h3>{{ title }}</h3>

<form action="" class="container">

    User Name: <input type="text" name="x" id="" [ngClass]="{'form-control':true, 'is-invalid':true}" [(ngModel)]="username" >

    Password: <input type="password" name="y" id="" class="form-control" [(ngModel)]="password">

    <br>

    <input type="button" value="Login" class="btn btn-success" (click)="fnLogin()"> &nbsp;

    <input type="reset" value="Reset" class="btn btn-warning">

</form>

</div>

Activty:

ngSwitch

Reactive Forms module

So far, we have done an approach called “Template driven forms”

Now, we are going to learn Reactive Forms

Steps:

1. In app.module.ts

We need to import “ReactiveFormsModule”

1. We should start from .ts file before .html
2. Create a component “product”

ng g c product

1. Go to product.component.ts file

And create an object representation of productForm

Inside the class, we have declared

productForm:any;

1. Now, lets define what are the members of productForm

In constructor, we need to inject a form builder

constructor(private fb:FormBuilder) { }

1. Inside the constructor, continue to define the form
2. export class ProductComponent implements OnInit {
3. productForm:any;

  constructor(private fb:FormBuilder) {

    this.productForm=this.fb.group(

      {

        id:[''],

        name:[''],

        price:['']

      }

    );

  }

8) go to product.component.html and define the form

<form [formGroup]="productForm">

Id: <input type="number" name="" id="" formControlName="id" class="form-control">

Name: <input type="text" name="" id="" formControlName="name" class="form-control">

Price: <input type="number" name="" id="" formControlName="price" class="form-control">

<br>

<input type="button" value="Add" class="btn btn-success">

</form>

<br>

{{ productForm.value | json}}

Task:

Below the product form, display a table of all products

So far, there are no products. So, declare an array of products in the .ts file

Whenever add button is pressed, a product object must be added to the products array in the .ts file.

For each product present in the products array, the table rows must be rendered.

productForm.controls.id.value

productForm.controls[“id”].value

import { Component, OnInit } from '@angular/core';

import { FormBuilder } from '@angular/forms';

import { Product } from '../product';

@Component({

  selector: 'app-product',

  templateUrl: './product.component.html',

  styleUrls: ['./product.component.css']

})

export class ProductComponent implements OnInit {

  productForm:any;

  products:Product[]=[];

  constructor(private fb:FormBuilder) {

    this.productForm=this.fb.group(

      {

        id:[''],

        name:[''],

        price:['']

      }

    );

  }

  ngOnInit(): void {

  }

  fnAdd()

  {

    // alert(JSON.stringify(this.productForm.value));

    var product=this.productForm.value;

    this.products.push(product);

    console.log(this.products);

  }

}

<form [formGroup]="productForm">

Id: <input type="number" name="" id="" formControlName="id" class="form-control">

Name: <input type="text" name="" id="" formControlName="name" class="form-control">

Price: <input type="number" name="" id="" formControlName="price" class="form-control">

<br>

<input type="button" value="Add" class="btn btn-success" (click)="fnAdd()">

</form>

<br>

{{ productForm.value | json}}

<table class="table table-bordered table-striped table-hover">

    <thead>

        <tr>

            <th>Id</th><th>Name</th><th>Price</th><th></th>

        </tr>

    </thead>

    <tbody>

        <tr \*ngFor="let p of products">

            <td>{{p.id}}</td>

            <td>{{p.name}}</td>

            <td>{{p.price}}</td>

        </tr>

    </tbody>

</table>

Task:

Place a “Delete” button and on clicking that button, the last product should be removed

Task:

Now, each row in products table should have a delete button. On clicking the button, the corresponding product (based on its id), should be removed.

<form [formGroup]="productForm">

Id: <input type="number" name="" id="" formControlName="id" class="form-control">

Name: <input type="text" name="" id="" formControlName="name" class="form-control">

Price: <input type="number" name="" id="" formControlName="price" class="form-control">

<br>

<input type="button" value="Add" class="btn btn-success" (click)="fnAdd()">

</form>

<br>

{{ productForm.value | json}}

<table class="table table-bordered table-striped table-hover">

    <thead>

        <tr>

            <th>Id</th><th>Name</th><th>Price</th><th></th>

        </tr>

    </thead>

    <tbody>

        <tr \*ngFor="let p of products">

            <td>{{p.id}}</td>

            <td>{{p.name}}</td>

            <td>{{p.price}}</td>

            <td>

                <input type="button" value="Delete" (click)="fnDelete(p.id)">

            </td>

        </tr>

    </tbody>

</table>

--------------ts

  fnDelete(id:number)

  {

// alert(id)

    var temp:Product[]=[];

    for(var x of this.products)

    {

      if(x.id!=id)

        temp.push(x);

    }

    this.products=temp;

  }

Task:

Put select button in every row of the table. On clicking that select button, that row should be patched to the form.

  fnSelect(id:number)

  {

    //how to patch an object to a reactive form?

    // console.log(id);

    for(let p of this.products)

    {

      var product:any;

      if(p.id==id)

      {

        // console.log("Found");

        // console.log(p);

        product=p;

      }

    }

    // console.log(product);

    this.productForm.patchValue(product);

  }