JS

1. Обмен данными между компонентами
   1. Получение данных в компоненте

Входные свойства компонента декорируются @Input и используются для получения данных от родительского компонента.

Например, создадим входной параметр под именем worker, которым будет являться объект с полями name, type и role.

@Input() worker: { name: string, type: string, role: string };

* 1. Отправка данных из компонента

Компоненты Angular могут отправлять пользовательские события, используя объект EventEmitter. Эти события можно обрабатывать в компоненте. Данный объект представляет собой подкласс Subject (реализован в библиотеке RxJS), который может быть, как наблюдаемым потоком, так и наблюдателем. Другими словами, EventEmitter может отправлять пользовательские события с помощью метода emit(), а также работать с наблюдаемыми потоками, задействую метод subscribe().

Например, создадим событие в компоненте под названием workerCreated, отправляющее объект с полями

@Output() workerCreated = new EventEmitter<{ name: string, role: string }>();

Для того, чтобы отправить событие в родительский компонент, нужно применить метод emit, в параметрах которого указать передаваемый объект.

this.workerCreated.emit({

name: nameInput.value,

role: this.workerRoleInput.nativeElement.value

});

Рассмотрим взаимодействие компонентов на примере приложения “Workers”. Приложение состоит из 3х компонентов:

Родительского AppComponent и 2х дочерних (вложенных) InterfaceComponent и WorkerElementComponent.

*\*Для создания нового компонента удобно использовать команду angular cli*

*ng generate component path/name*

*Например, для генерации вышеперечисленных компонентов были использованы команды*

*ng g c interface*

*ng g workerElement*

app.component.ts

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

@Component({

selector: 'app-root',

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

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

})

export class AppComponent {

workerElements = []; *// массив объектов, отобращаемых при помощи компонента worker*

onWorkerAdded(workerData: { name: string, role: string }) { *// обработчик события workerCreated из компонента Interface*

this.workerElements.push({

name: workerData.name,

role: workerData.role,

type: 'сотрудник'

});

}

onHeadWorkerAdded(workerData: { name: string, role: string }) { *// обработчик события headWorkerCreated из компонента Interface*

this.workerElements.push({

name: workerData.name,

role: workerData.role,

type: 'руководитель'

});

}

}

app.component.html

<div class="container">

<app-interface

(workerCreated)="onWorkerAdded($event)"

(headWorkerCreated)="onHeadWorkerAdded($event)"></app-interface> *<!--Задаем оработчики событий для EventEmmitorов из компонента interface -->*

<hr>

<div class="row">

<div class="col-xs-12">

<app-worker-element \*ngFor="let workerElement of workerElements"

[worker]="workerElement"></app-worker-element> *<!--Передача данных в свойство worker компонента workerElement -->*

</div>

</div>

</div>

interface.component.ts

import { Component, ElementRef, EventEmitter, Output, ViewChild } from '@angular/core';

@Component({

selector: 'app-interface',

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

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

})

export class InterfaceComponent {

@Output() workerCreated = new EventEmitter<{ name: string, role: string }>();

@Output() headWorkerCreated = new EventEmitter<{ name: string, role: string }>();

@ViewChild('workerRoleInput') workerRoleInput: ElementRef;

onAddWorker(nameInput: HTMLInputElement) {

this.workerCreated.emit({

name: nameInput.value,

role: this.workerRoleInput.nativeElement.value

});

}

onAddHeadWorker(nameInput: HTMLInputElement) {

this.headWorkerCreated.emit({

name: nameInput.value,

role: this.workerRoleInput.nativeElement.value

});

}

}

interface.component.html

<div class="row">

<div class="col-xs-12">

<div class="alert alert-info"><h1>Добавьте сотрудника или руководителя отдела</h1></div>

<label>Имя</label>

<input type="text" class="form-control" #workerNameInput>

<label>Отдел</label>

<input type="text" class="form-control" #workerRoleInput>

<br>

<button class="btn btn-primary btn-lg" (click) = "onAddWorker(workerNameInput)">Добавить сотрудника</button>

<button class="btn btn-primary btn-lg" (click) = "onAddHeadWorker(workerNameInput)">Добавить руководителя</button>

</div>

</div>

worker-element.component.ts

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

@Component({

selector: 'app-worker-element',

templateUrl: './worker-element.component.html',

styleUrls: ['./worker-element.component.css']

})

export class WorkerElementComponent {

@Input() worker: { name: string, type: string, role: string };

constructor() { }

}

worker-element.component.html

<div class="panel panel-default">

<div class="panel-heading">{{worker.name}}</div>

<div class="panel-body">

<p>

<strong \*ngIf="worker.type === 'сотрудник'">{{worker.role}}</strong>

<strong \*ngIf="worker.type === 'руководитель'" style="color:RGBA(255,0,0,1)">{{worker.role}}</strong>

</p>

</div>

</div>

Задание: Проект, приложенный к работе дополнить функционалом:

Добавить возможность удалять записи

Добавить возможность редактировать записи.

Сделать невозможным добавить запись с пустыми полями.