**Фреймворк Angular 9**

**Лабораторная работа № 2**

**Работа с компонентами #2**

**Задача. Оживить форму добавления постов. Научиться передавать данные из дочернего компонента в родительский. При добавлении поста отображать сразу его на сайте и очищать форму добавления. Организовать доступ к HTML элементам.**

**Ход работы**

1. В файл app.module.ts необходимо импортировать модуль с названием FormsModule. Он необходим для создания «two way binding».

import {FormsModule} from "@angular/forms";

@NgModule({  
 declarations: [  
 AppComponent,  
 PostFormComponent,  
 PostComponent  
 ],  
 imports: [  
 BrowserModule,  
 FormsModule  
 ],  
 providers: [],  
 bootstrap: [AppComponent]  
})  
export class AppModule { }

1. В файле post-from.component.ts заведем переменные title, text которые будут связаны с полями формы и также создадим метод addPost() для обработки нажатия кнопки добавления постов.

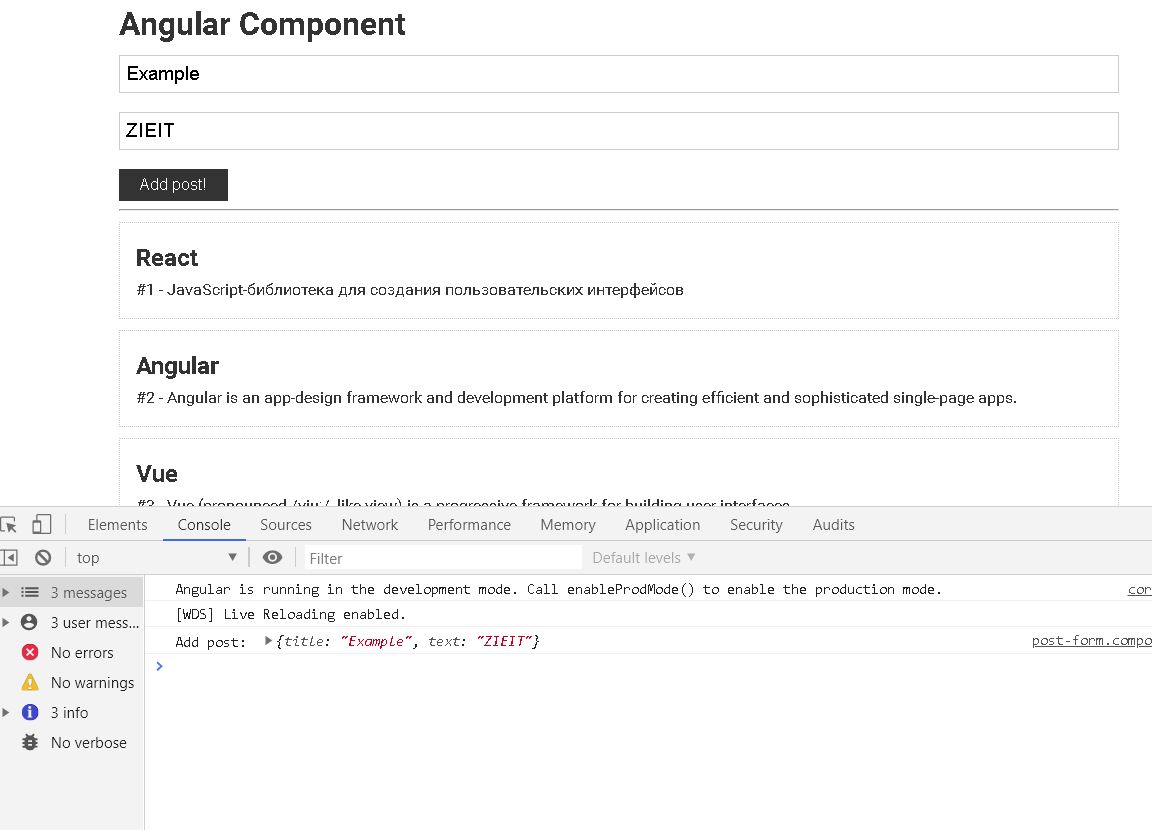
import {Post} from "../app.component";

export class PostFormComponent implements OnInit {  
title=''  
 text = ''  
 constructor() { }  
  
 ngOnInit() {  
 }  
  
 addPost() {  
 if(this.text.trim() && this.title.trim()) {  
 const post : Post ={  
 title: this.title,  
 text: this.text  
 }  
 console.log('Add post: ', post)  
 this.text=''  
 this.title=''  
 }  
 }  
}

В шаблоне post-from.component.html свяжем поля с созданными переменными используя концепт «two way binding» [(ngModel)] и обычный бандинг для кнопки с методом addPost()

<div>  
 <input type**="text"** class**="form-control"** placeholder**="Title..."** [(ngModel)]**="**title**"** >  
 <input type**="text"** class**="form-control"** placeholder**="Text..."** [(ngModel)]**="**text**"**>  
 <button class**="btn"** (click)**="**addPost()**"**>**Add post!**</button>  
  
</div>

В результате при нажатии на кнопку форма должна очищаться а в консоли отображаться созданный объект.



1. Теперь создадим процедуру передачи данных, с дочернего компонента родительскому, для добавления на страницу нового поста. Для этого в файле post-form.component.ts внесем следующую информацию.

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

import {Post} from "../app.component";

export class PostFormComponent implements OnInit {  
title=''  
 text = ''  
 @Output() addPostUser: EventEmitter<Post> = new EventEmitter<Post>()  
 constructor() { }  
  
 ngOnInit() {  
 }  
  
 addPost() {  
 if(this.text.trim() && this.title.trim()) {  
 const post : Post ={  
 title: this.title,  
 text: this.text  
 }  
 this.addPostUser.emit(post)  
this.text=''  
 this.title=''  
 }  
 }  
}

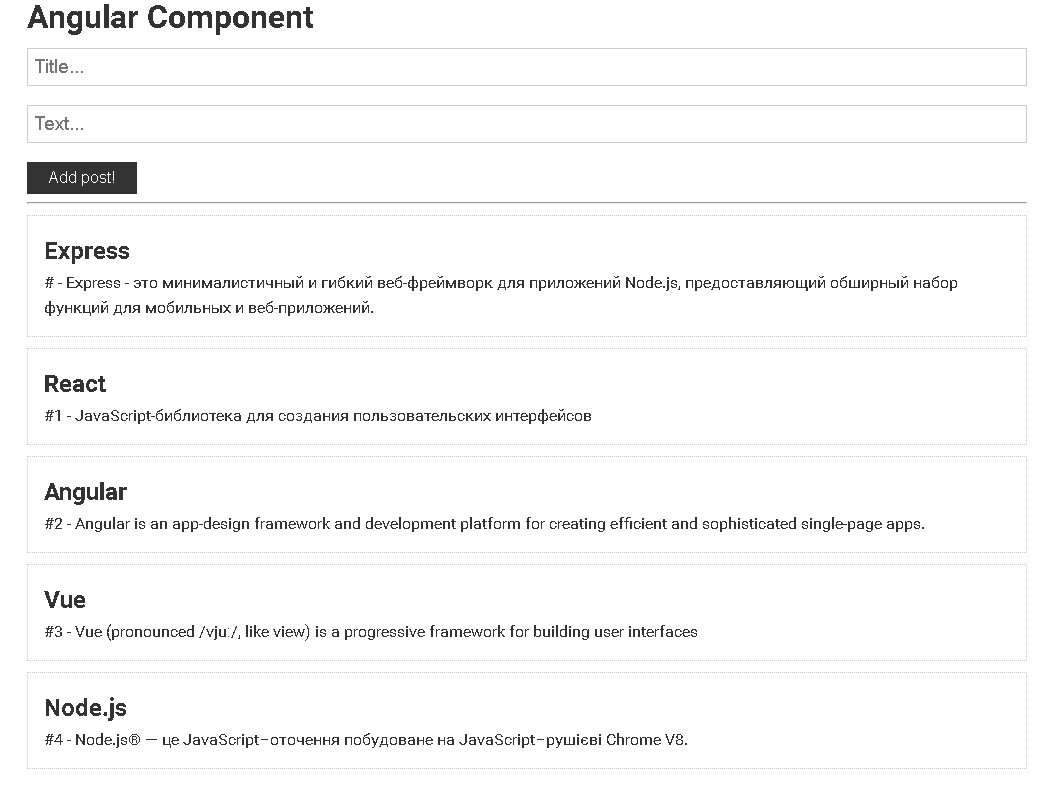
1. Теперь в родительском шаблоне можно обработать полученный имит. Добавим прослушку события из шаблона с тем же именем (addPostUser) и передадим его в метод updatePosts

<div class**="container"**>  
 <h1>**Angular Component**</h1>  
 <app-post-form  
 (addPostUser)**="**updatePosts(*$event*)**"** >  
  
 </app-post-form>  
 <hr />  
 <app-post *\*ngFor***="**let *p* of posts**"** [post]**="***p***"**></app-post>  
  
</div>

1. В файле app.component.ts реализуем метод updatePosts на добавление нового поста в начало массива.

export class AppComponent {  
posts: Post[] = [  
 {title:"React", text: "JavaScript-библиотека для создания пользовательских интерфейсов", id: 1},  
 {title:"Angular", text: "Angular is an app-design framework and development platform for creating efficient and sophisticated single-page apps.", id: 2},  
 {title:"Vue", text: "Vue (pronounced /vjuː/, like view) is a progressive framework for building user interfaces", id: 3},  
 {title:"Node.js", text: "Node.js® — це JavaScript–оточення побудоване на JavaScript–рушієві Chrome V8.", id: 4}  
]  
 updatePosts (event){  
  
 this.posts.unshift(event)  
 }  
}

Проверим полученный результат:



**!!!Самостоятельно. При добавлении нового поста, id должен быть равным 1. Все остальные посты пересчитать с учетом добавленного.**

1. Добавим в приложение кнопку по нажатию на которую в поля title и text будут загружаться различные стили. А также заведем локальные референции для полей ввода через #

Для начала отредактируем шаблон post-form.component.html и добавим саму кнопку

<div>  
 <input type**="text"** class**="form-control"** #*myInputText* placeholder**="Title..."** [(ngModel)]**="**title**"** >  
 <input type**="text"** class**="form-control"** #*myInputTitle* placeholder**="Text..."** [(ngModel)]**="**text**"**>  
 <button class**="btn"** (click)**="**addPost()**"**>**Add post!**</button>  
<button class**="btn"** (click)**="**onLoadDefault()**"**>**Default style**</button>  
</div>

1. Установим специальный декоратор @ViewChild для доступа к DOM элементам. Также в файле post-form.component.ts топишем логику функции onLoadDefault()

export class PostFormComponent implements OnInit {  
 title=''  
 text = ''  
 styleToggle=false  
 @Output() addPostUser: EventEmitter<Post> = new EventEmitter<Post>()  
 @ViewChild('myInputText',{static: false}) myinputText: ElementRef  
 @ViewChild('myInputTitle',{static: false}) myinputTitle: ElementRef  
 constructor() { }  
  
 ngOnInit() {  
 }  
  
 addPost() {  
 if(this.text.trim() && this.title.trim()) {  
 const post : Post ={  
 title: this.title,  
 text: this.text  
 }  
 this.addPostUser.emit(post)  
this.text=''  
 this.title=''  
 }  
 }  
  
 onLoadDefault () {  
 this.styleToggle=!this.styleToggle  
 if(this.styleToggle) {  
 this.myinputText.nativeElement.style.color = "red"  
 this.myinputTitle.nativeElement.style.fontWeight = "bold"  
 } else {  
 this.myinputText.nativeElement.style.color = 'black'  
 this.myinputTitle.nativeElement.style.fontWeight = "normal"  
  
 }  
  
 }  
  
}

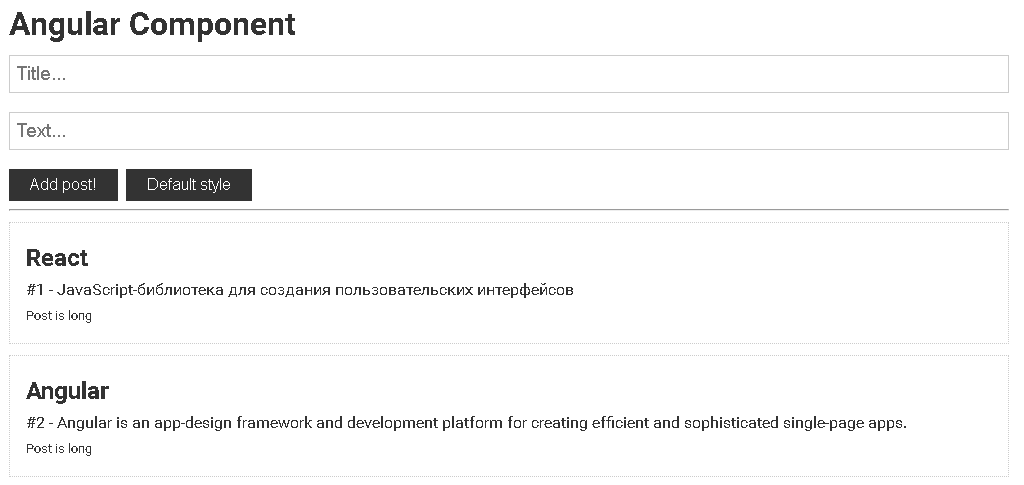
1. Рассмотрим еще одну возможность передачи HTML в компонент. Выведем информацию в каждом посте о том короткая информация или длинная.

Для этого отредактируем app.cpmponent.html

<app-post *\*ngFor***="**let *p* of posts**"** [post]**="***p***"**>  
<small *\*ngIf***="***p*.text.length>10; else *small***"**>**Post is long**</small>  
<ng-template #*small* >  
 <small> **Post is short**</small>  
</ng-template>  
  
 </app-post>

Для того чтобы эта информация отображалась в шаблоне компонента PostComponent нужно добавить специальный компонент в файл post.component.html

<ng-content> </ng-content>



1. Создадим кнопку для удаления постов. Для этого в post.component.html добавим кнопку вызывающую метод removePost().

<button class**="btn"** (click)**="**removePost()**"**>**&#10006;**</button>

В post.component.ts реализуем метод для передачи id поста главному компоненту.

export class PostComponent implements OnInit {  
@Input() post: Post  
 @Output() postid: EventEmitter<number> = new EventEmitter<number>()  
 removePost() {  
 this.postid.emit(this.post.id)  
 }  
 ngOnInit() {  
 }  
}

После чего повесим прослушку метода postid в шаблоне app.component.html

<app-post *\*ngFor***="**let *p* of posts**"** [post]**="***p***"** (postid)**="**removeIdPost(*$event*)**"** >

И реализуем метод removeIdPost() в файле app.component.ts

removeIdPost(id:number) {  
this.posts = this.posts.filter((item)=>item.id!=id)  
}

В итоге получим такой результат:

