JS

Лабораторная работа 6.

1. **Компоненты в angular.**

Компонент - это основной элемент Angular-приложения. Каждый из них состоит из двух частей: представления, определяющего пользовательский интерфейс, и класса, реализующего логику, лежащую за представлением.

@Component({

selector: 'app-root',

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

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

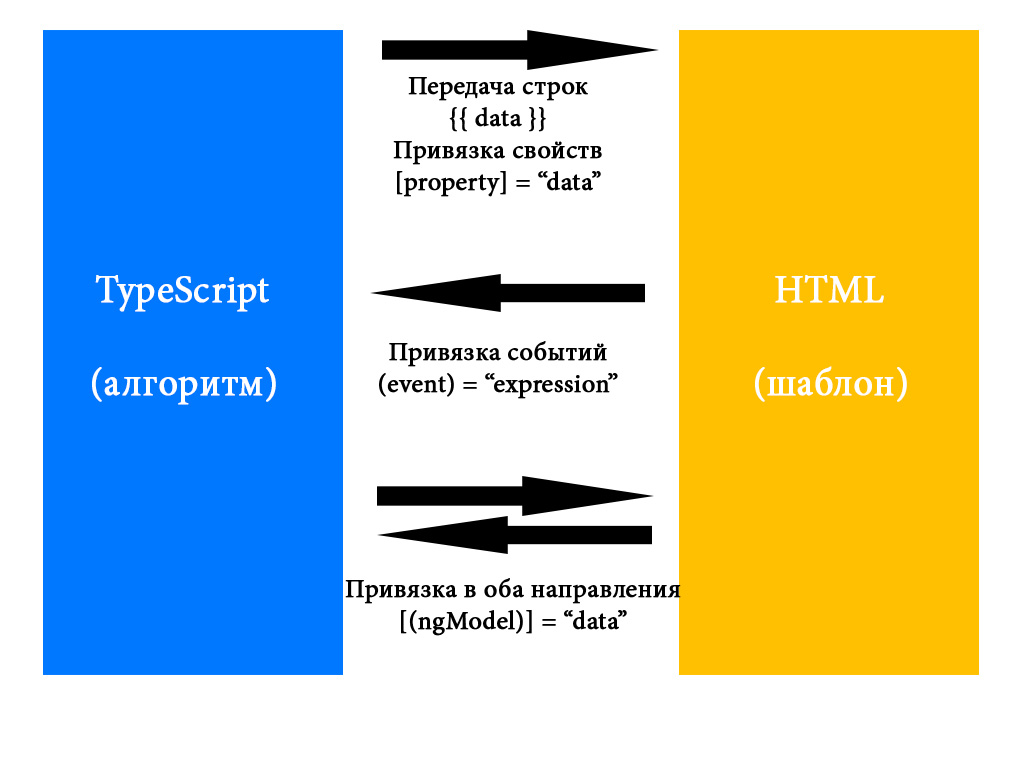
})

export class AppComponent {

}

К классу компонента прикреплен декоратор @Component, который содержит

* селектор (идентификатор компонента в приложении),
* шаблон - представление(или ссылку на шаблон),
* массив стилей css.
  1. Обмен данными внутри компонента



* Передача строк

<h1>{{str}}</h1>

* Привязка свойств

<button [disabled]="btnDisabled">Кнопка</button>

* Привязка событий

<div class="my-block" (mouseover)="onOver($event)">Блок</div>

* Привязка в оба направления

<input type="text" [(ngModel)]="str">

Для использования привязки в оба направления необходимо импортировать FormsModule.

@NgModule({

declarations: [

AppComponent

],

imports: [

BrowserModule,

FormsModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

* 1. Локальная референция

– это сопоставление элемента DOM с именем внутри шаблона.

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

<button (click)="onMyClick(someNameInput)">Кнопка</button>

onMyClick(elm: HTMLInputElement) {

console.log(elm);

}

Получить доступ к элементу DOM внутри шаблона можно при помощи декоратора @ViewChild

@ViewChild('someNameInput') someNameInput: ElementRef;

* 1. Жизненный цикл компонента

В процессе создания/работы/уничтожения компонент вызывает ряд функций. При создании экземпляра компонента первым вызывается конструктор, но во время его работы свойства еще не инициализированы. После завершения работы конструктора Angular вызовет следующие функции обратного вызова, если вы их реализовали.

**ngOnChanges**: вызывается до метода ngOnInit() при начальной установке свойств, которые связаны механизмом привязки, а также при любой их переустановке или изменении их значений. Данный метод в качестве параметра принимает объект класса SimpleChanges, который содержит предыдущие и текущие значения свойства.

**ngOnInit**: вызывается один раз после установки свойств компонента, которые участвуют в привязке. Выполняет инициализацию компонента. Запускается после конструктора.

**ngDoCheck**: вызывается при каждой проверке изменений свойств компонента сразу после методов ngOnChanges и ngOnInit

**ngAfterContentInit**: вызывается один раз после метода ngDoCheck() после вставки содержимого в представление компонента кода html

**ngAfterContentChecked**: вызывается фреймворком Angular при проверке изменений содержимого, которое добавляется в представление компонента. Вызывается после метода ngAfterContentInit() и и после каждого последующего вызова метода ngDoCheck().

**ngAfterViewInit**: вызывается фреймворком Angular после инициализации представления компонента, а также представлений дочерних компонентов. Вызывается только один раз сразу после первого вызова метода ngAfterContentChecked()

**ngAfterViewChecked**: вызывается фреймворком Angular после проверки на изменения в представлении компонента, а также проверки представлений дочерних компонентов. Вызывается после первого вызова метода ngAfterViewInit() и после каждого последующего вызова ngAfterContentChecked()

**ngOnDestroy**: вызывается перед тем, как фреймворк Angular удалит компонент.

export class AppComponent implements OnInit {

constructor() {

console.log('constructor');

}

ngOnInit() {

console.log('onInit');

}

}

* 1. Оператор условия в шаблоне

В шаблоне компонента возможно использование директивы ngIf для реализации функционала оператора условия.

<div class="my-block" \*ngIf="showBlock == true">Блок</div>

Возможна реализация функционала if else

<div class="my-block" \*ngIf="showBlock == true; else elseBlock">Блок</div>

<ng-template #elseBlock>

<div class="my-block">Другой блок</div>

</ng-template>

* 1. Цикл в шаблоне

numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9];

В шаблоне возможен вывод данных в цикле. Реализуется при помощи директивы ngFor.

<p \*ngFor="let number of numbers; let i = index, let last = last, let first = first">

<span \*ngIf="first">Первый </span><span \*ngIf="last">Последний </span>{{i}} - {{number}}

<span \*ngIf="!first">Предыдущий элемент {{numbers[i - 1]}}</span>

</p>

* 1. Возможно динамическое изменение стилей при помощи атрибута style и директивы ngStyle

<div [style.font-size]="isSpecial ? 'x-large' : 'smaller'">

Большой или маленький текст

</div>

<div class="my-block" [ngStyle]="{'height': myHeight}">Блок</div>

* 1. Динамической изменение css классов осуществляется при помощи директивы ngClass

<p [ngClass]="{'green-color': isGreen == true}">Датчик номер </p>

<https://angular.io/guide/template-syntax>

1. Bootsrap

Для удобства можно подключить css framework bootstrap. Выполним команду

npm install bootstrap --save

И добавим файл стилей bootstrap.min.css в angular.json

"styles": [

"node\_modules/bootstrap/dist/css/bootstrap.min.css",

"src/styles.css"

],

1. Задание

Создать приложение на базе одного компонента. В приложении используются датчики, у датчика есть свойства

online: boolean;

name: string;

В начале работы приложения есть массив из датчиков

[датчик 1, датчик 2…]

свойство online генерируется случайным образом.

Датчики должны быть отображены на странице, датчики со свойством online равным true отображаются зеленым цветом, другие красным.

Должна быть предусмотрена возможность добавления датчика (свойство online указывается вручную) и удаления датчиков из списка (кнопкой удалить).