**Basic Commands**

1. ng new app-one : Create new project
2. ng serve : Start project
3. ng g c : create component

**Data binding**

Output data from TS -> HTML :

**String interpolation** {{var\_name}} -> Only works innerHTML

**Property Binding** [property]=”data” -> Change properties/attributes of HTML element

[disabled]=”variable”

React to user event from HTML -> TS :

**Event binding** (event)=”expression”

(click)=”anyMethod()”

(click)=”anyMethod(‘any param’)”

(click)=”anyMethod(‘any param’ , $event)”

(click)=”anyMethod($event)”

In ts file -> anyMethod(event:any){

Console.log(event); // here you will see what type of event

// Change the any to MouseEvent in param

const clicked = event.target as HTMLElement;

this.whereClicked = clicked.innerText;

}

**Two way binding**: [(ngModel)]=”data”

![Graphical user interface, application

Description automatically generated]()

**Find what type of element:**

console.dir(document.querySelector(‘.btn’));

scroll down to \_\_proto

you will see then. EG HTMLInputElement , HTMLButtonElement

**Find Properties and Events of an element**:

* In chrome console -> console.dir(document.querySelector(‘.btn’));
* Or ![Graphical user interface, application, Teams

  Description automatically generated]()
* Or google YOUR\_ELEMENT properties or YOUR\_ELEMENT events

<https://www.w3schools.com/jsref/dom_obj_event.asp>

**Directives**

1. **Structural directives**—change the DOM layout by adding and removing DOM elements.

\*ngIf:

<div **\*[ngIf](https://angular.io/api/common/NgIf)**="hero">This is if condition, value can be boolean</div>

With Else:

<div **\*[ngIf](https://angular.io/api/common/NgIf)**="hero; else localRef">This is if condition, value can be boolean</div>

<ng-template #localRef >

<div>this is else condition, text of #localref can be any</div>

</ ng-template>

NgFor—repeat a node for each item in a list.

<div \*[ngFor](https://angular.io/api/common/NgForOf)="let item of items; let i=index"></div>

1. **Attribute directives**-change the appearance or behavior of an element, component, or another directive.

ngStyle: Set many inline styles simultaneously and dynamically

<div [[ngStyle](https://angular.io/api/common/NgStyle)]="currentStyles">will accept an object</div>

currentStyles: {};

this.currentStyles = {

'font-style': this.canSave ? 'italic' : 'normal',

'font-weight': !this.isUnchanged ? 'bold' : 'normal',

'font-size': this.isSpecial ? '24px' : '12px' };

}

NgClass—adds and removes a set of CSS classes based on condition. Use property binding if only one class needs to be added/removed (like toggle)

Accepts an object. Key is name of the class, value is true or false.

<li [ngClass]="{

'text-success':person.country === 'UK',

'text-primary':person.country === 'USA',

'text-danger':person.country === 'HK'

}">{{ person.name }} ({{ person.country }})

</li>

**Component and Databinding**

**Component Property Binding:**

From **PARANT** to **CHILD** -> **@Input()**

**-> @Input(‘alias’)**

Good doc: https://angular.io/guide/inputs-outputs

**Component Event Binding:**

From **CHILD** to **PARENT** -> **@Output()**

**-> @Output(‘alias’)**

Good doc: <https://angular.io/guide/inputs-outputs>

**View Encapsulation**

**To change angulars default CSS rendering**

**Local Reference in template/html**

**To assign a reference to any html element and use it only in template.**

The only way to get this ref in ts file is from a method param mentioned

In html template.

<input type="text"

class="form-control"

#serverInputElement>

<button

class="btn btn-primary"

(click)="onAddServer(serverInputElement)">Add Server</button>

onAddServer(serverInputElement:HTMLInputElement) {

this.serverCreated.emit(

{name: serverInputElement.value.trim() ,

content: this.newServerContent,

}

);

<https://angular.io/guide/template-reference-variables>

**Access to the Template & DOM with @ViewChild**

**To access template and dom element in ts file**

<input

type="text"

class="form-control"

#contentInputElement>

@ViewChild('contentInputElement' , {static : true}) contentInputElement:ElementRef;

onAddServer(serverInputElement:HTMLInputElement) {

this.serverCreated.emit(

{name: serverInputElement.value.trim() ,

content: this.contentInputElement.nativeElement.value

}

);

**ng-content**

**Angular Lifecycle Hook**

Graphical user interface, text, application

Description automatically generated

**Custom Directive (Better way to access DOM)**

<p appHighlight [defaultColor]="'green'" [highlightColor]="'blue'">this is custom directive</p>

@Directive({

selector: '[appHighlight]'

})

export class HighlightDirective implements OnInit{

@Input() defaultColor:string = "transparent";

@Input() highlightColor:string = "blue";

constructor(private elRef:ElementRef, private renderer:Renderer2) { }

ngOnInit() {

this.renderer.setStyle(this.elRef.nativeElement , 'background-color' , this.defaultColor);

}

@HostListener('mouseenter') mouseEnter(event: Event){

this.renderer.setStyle(this.elRef.nativeElement , 'background-color' , this.highlightColor);

}

@HostListener('mouseleave') mouseleave(event: Event){

this.renderer.setStyle(this.elRef.nativeElement , 'background-color' , this.defaultColor);

}

}

**Services**

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

@Injectable({

providedIn: 'root'

})

export class AccountsService {

accounts = [

{

name: 'Master Account',

status: 'active'

},

{

name: 'Testaccount',

status: 'inactive'

},

{

name: 'Hidden Account',

status: 'unknown'

}

];

constructor() { }

createAccount(name:string , status:string){

this.accounts.push({name: name , status: status});

}

updateAccount(id: number , status: string){

this.accounts[id].status = status;

}

}

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

import { AccountsService } from '../services/accounts.service';

import { LoggingService } from '../services/logging.service';

@Component({

selector: 'app-new-account',

templateUrl: './new-account.component.html',

styleUrls: ['./new-account.component.css']

})

export class NewAccountComponent {

constructor(private accountsService: AccountsService){}

onCreateAccount(accountName: string, accountStatus: string) {

this.accountsService.createAccount(accountName , accountStatus);

}

}

<div class="form-group">

<label>Account Name</label>

<input

type="text"

class="form-control"

#accountName>

</div>

<div class="form-group">

<select class="form-control" #status>

<option value="active">Active</option>

<option value="inactive">Inactive</option>

<option value="hidden">Hidden</option>

</select>

</div>

<button

class="btn btn-primary"

(click)="onCreateAccount(accountName.value, status.value)">

Add Account

</button>

Another Example of passing data between components

Service :

import { EventEmitter, Injectable } from '@angular/core';

import { Recipe } from './recipe.model';

@Injectable({

providedIn: 'root'

})

export class RecipeService {

private recipes: Recipe[] = [

new Recipe('Detox Moroccan-Spiced Chickpea Glow Bowl' , 'Sweats, thick socks, and a big glow bowl loaded with Moroccan-spiced chickpeas' , 'https://pinchofyum.com/wp-content/uploads/Moroccan-Chickpea-Bowls-4.jpg'),

new Recipe('Spicy Shrimp Tacos with Garlic Cilantro Lime Slaw' , 'Spice-loaded shrimp tucked in between smashed avocado and a cabbage slaw that is heavy with a homemade creamy lime sauce' , 'https://pinchofyum.com/wp-content/uploads/Shrimp-Tacos-with-Slaw.jpg'),

new Recipe('The Best Protein Pancakes' , 'With peanut butter and chocolate chips and a little bit of maple syrup' , 'https://pinchofyum.com/wp-content/uploads/Protein-Pancakes-1-4.jpg')

];

selectedRecipe = new EventEmitter<Recipe>();

constructor() { }

getAllRecipes():Recipe[]{

return [...this.recipes];

}

}

A component that emits/triggers some change

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

import { Recipe } from '../../recipe.model';

import { RecipeService } from '../../recipe.service';

@Component({

selector: 'app-recipe-item',

templateUrl: './recipe-item.component.html',

styleUrls: ['./recipe-item.component.css']

})

export class RecipeItemComponent implements OnInit {

@Input() recipe:Recipe; // ignore this line

constructor(private recipeService: RecipeService) { }

ngOnInit(): void {

}

onRecipeClicked(){

this.recipeService.selectedRecipe.emit(this.recipe);

}

}

A component that’s listing/subscribing to any trigger/emit

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

import { Recipe } from '../recipe.model';

import { RecipeService } from '../recipe.service';

@Component({

selector: 'app-recipe-details',

templateUrl: './recipe-details.component.html',

styleUrls: ['./recipe-details.component.css']

})

export class RecipeDetailsComponent implements OnInit {

selectRecipe: Recipe;

constructor(private recipeService:RecipeService) { }

ngOnInit(): void {

this.recipeService.selectedRecipe.subscribe(

(recipe : Recipe) => {

this.selectRecipe = recipe;

}

);

}

}