# **Youtube - Angular 6 Tutorial**

Link: <https://www.youtube.com/watch?v=0eWrpsCLMJQ&list=PLC3y8-rFHvwhBRAgFinJR8KHIrCdTkZcZ&index=1>

Tartalom

[**Youtube - Angular 6 Tutorial** 1](#_Toc527628414)

[**1.** **– Introduction** 1](#_Toc527628415)

[**2.** **– Getting Started** 1](#_Toc527628416)

[**3.** **– Hello World App** 1](#_Toc527628417)

[**4.** **– Components** 2](#_Toc527628418)

[**5.** **– Interpolation** 4](#_Toc527628419)

[**6.** **– Property Binding** 5](#_Toc527628420)

[**7.** **– Class binding** 6](#_Toc527628421)

[**8.** **– Style Binding** 6](#_Toc527628422)

[**9.** **– Event Binding** 7](#_Toc527628423)

[**10.** **– Template Reference Variables** 8](#_Toc527628424)

[**11.** **– Two Way Binding** 9](#_Toc527628425)

[**12.** **– Structural Directives - ngIF** 10](#_Toc527628426)

[**13.** **– Structural Directives – ngswitch** 12](#_Toc527628427)

[**14.** **– Structural Directives – NgFor** 13](#_Toc527628428)

[**15.** **– Component Interaction** 14](#_Toc527628429)

[**16.** **– Pipes** 14](#_Toc527628430)

[**17.** **– Services** 14](#_Toc527628431)

[**18.** **– Depedency Injection** 14](#_Toc527628432)

[**19.** **– Using a Service** 14](#_Toc527628433)

[**20.** **– HTTP and Observables** 14](#_Toc527628434)

[**21.** **– Fetch Data Using HTTP** 14](#_Toc527628435)

[**22.** **– HTTP Error Handling** 15](#_Toc527628436)

[**23.** **– Routing and navigation** 15](#_Toc527628437)

[**24.** **– Wildcard Route and Redirecting Routes** 15](#_Toc527628438)

[**25.** **– Route Parameters** 15](#_Toc527628439)

[**26.** **– ParamMap Observable** 15](#_Toc527628440)

[**27.** **– Optional Route Parameters** 15](#_Toc527628441)

[**28.** **– Relative Navigation** 15](#_Toc527628442)

[**29.** **– Child Routes** 15](#_Toc527628443)

[**30.** **–Angular 6 New Features and Changes** 15](#_Toc527628444)

[**31.** **– Updating your app to Angular 6** 15](#_Toc527628445)

## **– Introduction**

Angular 6 – April 2018

## **– Getting Started**

HTML, CSS, JS, Basics of Typescript

Angular CLI, VS CODE Text Editor

## **– Hello World App**

ng new hello-world

cd hello-world

ng serve

localhost:4200

Modules

Root Module – App Module

Module=Component + Services

Components

Navigation,Sidebar,Main Content

Root component

Angular app - one or more modules

Module - one or more components and services

Components - html + class

Services - Business logic

Root component {Component1, Component2, ...}

Module = Component + Services

## **– Components**

Template + Class + Metadata = Component

Template – View + HTML

Class – Code ,TypeScript, Data & Methods

Metadata – Information, Decorator

ng generate component test

3 way to specify the selector:

* custom html tag

@Component({

selector: 'app-test',

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

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

})

<app-test></app-test>

* class name

@Component({

selector: '.app-test',

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

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

})

<div class="app-test">

</div>

* as an attribute

@Component({

selector: '[app-test]',

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

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

})

<div app-test>

</div>

3 way to specify the templateUrl:

* templateUrl -> file

@Component({

selector: '[app-test]',

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

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

})

* template -> Inline

@Component({

selector: '[app-test]',

template: '<div>Inline template</div>',

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

})

* template -> Inline multiple line

@Component({

selector: '[app-test]',

template: `<div>

Multiple line Inline template

</div>`,

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

})

2 way to specify the styleUrls:

* styleUrls

@Component({

selector: 'app-test',

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

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

})

* styles -> inline

@Component({

selector: 'app-test',

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

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

styles: [`p {

color:red;

}`]

})

## **– Interpolation**

<p>Welcome {{ name }}</p>

<p>{{ 2 + 2 }}</p>

<p>{{ "Welcome" + name }}</p>

<p>{{ "name: " + name + ", length: " + name.length }}</p>

<p>{{ name.toUpperCase() + name.toLowerCase() }}</p>

<p>{{ greetUser()}}</p>

<p> a = 2+2 //ERR: Bindings cannot contain assignments </p>

<p>window.location.href //read property 'location' of undefined</p>

<p>{{ siteUrl }}</p>

public siteUrl = window.location.href;

public name = "Akira"; //or

name = 'Andrea';

export class TestComponent implements OnInit {

public name = "codevolution";

constructor() { }

ngOnInit() {

}

greetUser(){

return "Hello " + this.name;

}

}

\*\*\*html template\*\*\*

<p>Greet User: {{ greetUser() }}</p>

## **– Property Binding**

Attribute vs Property

Attributes and Properties are not the same.

Attributes - HTML - cannot change once there are initialized

Properties DOM (Document Object Model) - however can change

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

@Component({

selector: 'app-test',

template: `

<input type="text" value="Andrea">

<input type="text" [id] = "myId" value="Andrea">

<input [disabled] = "isDisabled" type="text" id = "{{myId}}" value="Andrea">

<input bind-disabled = "isDisabled" type="text" id = "{{myId}}" value="Andrea">`,

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

})

export class TestComponent implements OnInit {

public name = "Andrea";

public myId = "testId";

constructor() { }

ngOnInit() {

}

}

dev -> console:

$0.getAttribute('value')

$0.value

## **– Class binding**

export class TestComponent implements OnInit {

public name = "Codevolution";

public successClass = "text-success";

public hasError = false;

public isSpecial = true;

public messageClasses = {

"text-success": !this.hasError,

"text-danger": this.hasError,

"text-special": this.isSpecial

}

constructor() { }

ngOnInit() {

}

greetUser(){

return "Hello " + this.name;

}

}

\*\*\*html\*\*\*

<p class="text-danger">Greet User: {{ greetUser() }}</p>

<h2 class="text-success">Some text here 1 .... </h2>

<h2 [class]="successClass">Some text here 2 .... </h2>

<h2 class="text-special" [class]="successClass">Some text here 2 .... </h2>

<!-- Nem lehet mind a kettot hasznlani egyszerre -->

<h2 [class.text-danger]="hasError">Some text here 2 .... </h2>

<h2 [ngClass]="messageClasses">Some text here 2 .... </h2>

## **– Style Binding**

export class TestComponent implements OnInit {

public name = "Codevolution";

public hasError = false;

public isSpecial = true;

public highlightColor = "orange";

public titleStyles = {

color: "blue",

fontStyle: "italic"

}

// camel case -> fontStyle

constructor() { }

ngOnInit() {

}

}

\*\*\*html\*\*\*

<h2>Welcome {{ name }}</h2>

<h2 [style.color]="'orange'">Style Binding</h2>

<h2 [style.color]="hasError ? 'red' : 'green'">Style Binding</h2>

<h2 [style.color]="'orange'">Style Binding 2</h2>

<h2 [style.color]="highlightColor">Style Binding</h2>

<h2 [ngStyle]="titleStyles">Style Binding</h2>

## **– Event Binding**

Data Binding : Class -> Template

Event Binding: Template -> Class

@Component({

selector: 'app-test',

template: `

<h2>

Welcome {{ name }}

</h2>

<button (click)="onClick()">Greet</button>

<button (click)="onClickSecond($event)">Greet 2</button>

<button (click)="greeting='Welcome Vishwas'">Greet 3</button>

{{ greeting }}`,

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

})

export class TestComponent implements OnInit {

public name = "Codevolution";

public greeting = ""; //add a new property

constructor() { }

ngOnInit() {

}

onClick(){

console.log("Welcome to Codevolution");

this.greeting = "Welcome to Codevolution";

this.name = "Andrea";

}

onClickSecond(event){

console.log(event);

// MouseEvent {isTrusted: true, screenX: 1077, screenY: 202, clientX: 85, clientY: 87, …}

this.greeting = event.type;

//click

}

}

## **– Template Reference Variables**

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

@Component({

selector: 'app-test',

template: `

<h2>

Welcome {{ name }}

</h2>

<input #myInput type="text">

<button (click)="logMessage(myInput.value)">Log</button>

<button (click)="logMessage(myInput)">Log</button>`,

//Andrea, <input \_ngcontent-c1="" type="text">">

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

})

export class TestComponent implements OnInit {

public name = "Codevolution";

constructor() { }

ngOnInit() {

}

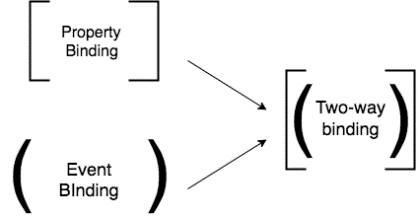
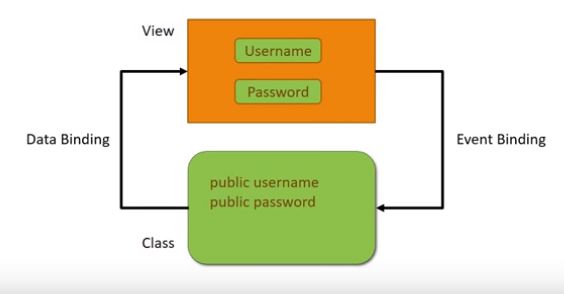
logMessage(value){

console.log(value);

}

}

## **– Two Way Binding**



\*\*\*app.module.ts\*\*\*

import { BrowserModule } from '@angular/platform-browser';

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

**import { FormsModule } from '@angular/forms';**

import { AppComponent } from './app.component';

import { TestComponent } from './test/test.component';

@NgModule({

declarations: [

AppComponent,

TestComponent

],

imports: [

BrowserModule,

**FormsModule**

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

\*\*\*test.component.ts\*\*\*

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

@Component({

selector: 'app-test',

template: `

<h2>

Welcome {{ name }}

</h2>

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

{{ name }}`,

// [()] banana in the box

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

})

export class TestComponent implements OnInit {

public name = "";

constructor() { }

ngOnInit() {

}

}

## **– Structural Directives - ngIF**

Add or remove HTML elements

* ngIf (render HTML element)
* ngSwitch (render HTML element)
* ngFor (render list element)

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

@Component({

selector: 'app-test',

template: `

<h2 \*ngIf="true">

Structural Directives

</h2>

<h2 \*ngIf="false">

structural Directives

</h2>

<h2 \*ngIf="displayName">

structural Directives

</h2>

<h2 \*ngIf="displayName; else elseBlock">

Some text 1

</h2>

<ng-template #elseBlock>

<h2>

Some text 2

</h2>

</ng-template>`,

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

})

export class TestComponent implements OnInit {

displayName = true;

constructor() { }

ngOnInit() {

}

}



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

@Component({

selector: 'app-test',

template: `

<div \*ngIf="displayName; then thenBlock; else elseBlock">

</div>

<ng-template #thenBlock>

<p>Then Block text</p>

</ng-template>

<ng-template #elseBlock>

<p>Else Block text</p>

</ng-template>`,

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

})

export class TestComponent implements OnInit {

displayName = true;

constructor() { }

ngOnInit() {

}

}

## **– Structural Directives – ngswitch**

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

@Component({

selector: 'app-test',

template: `

<div [ngSwitch]="color">

<div \*ngSwitchCase="'red'">You picked Red color</div>

<div \*ngSwitchCase="'green'red">You picked Green color</div>

<div \*ngSwitchCase="'blue'">You picked Blue color</div>

<div \*ngSwitchDefault>Pick again</div>

</div>

`,

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

})

export class TestComponent implements OnInit {

public color = "";

constructor() { }

ngOnInit() {

}

}

## **– Structural Directives – NgFor**

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

@Component({

selector: 'app-test',

template: `

<div \*ngFor="let color of colors; index as i">

<h2>{{ i }} : {{ color }} </h2>

</div>

<div \*ngFor="let color of colors; first as f">

<h2>{{ f }} : {{ color }} </h2>

</div>

<div \*ngFor="let color of colors; last as l">

<h2>{{ l }} : {{ color }} </h2>

</div>

<div \*ngFor="let color of colors; odd as o">

<h2>{{ o }} : {{ color }} </h2>

</div>

<div \*ngFor="let color of colors; even as e">

<h2>{{ e }} : {{ color }} </h2>

</div>

`,

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

})

export class TestComponent implements OnInit {

public colors = ["red", "blue", "green", "yellow"];

constructor() { }

ngOnInit() {

}

}

## **– Component Interaction**

## **– Pipes**

## **– Services**

## **– Depedency Injection**

## **– Using a Service**

## **– HTTP and Observables**

## **– Fetch Data Using HTTP**

## **– HTTP Error Handling**

## **– Routing and navigation**

## **– Wildcard Route and Redirecting Routes**

## **– Route Parameters**

## **– ParamMap Observable**

## **– Optional Route Parameters**

## **– Relative Navigation**

## **– Child Routes**

## **–Angular 6 New Features and Changes**

## **– Updating your app to Angular 6**