https://www.tutorialspoint.com/angular7/index.htm

# Angular

Egy nyílt forrású javascript keretrendszer.

| Version | Released Date |
|---|---|
| Angular JS | October 2010 |
| Angular 2.0 | Sept 2016 |
| Angular 4.0 | March 2017 |
| Angular 5.0 | November 2017 |
| Angular 6.0 | May 2018 |
| Angular 7.0 | October 2018 |
| Angular 8.0 | March/April 2019 |
| Angular 9.0 | September/ October 2019 |

Angular update:

```
ng update @angular/cli @angular/core
```

## Angular7 - Environment Setup

- Nodejs
- Npm
- Angular CLI
- IDE a forráskód írásához

Angular CLI install:

```
npm install –g @angular/cli
```

Új projekt létrehozása:

```
ng new my-dream-app // name of the project
cd my-dream-app
```

```
ng serve
```

**Alapértelmezetten:**

```
http://localhost:4200/
```

Megváltoztatása:

```
ng serve --host 0.0.0.0 -port 4205
```

The angular7-app/ folder has the following **folder structure**−

- **e2e/** − end to end test folder
- **node_modules/** − The npm package installed is node_modules.
- **src/** −Ahova írjuk az alkalmazás kódját. Az src/-n bekül az app/-ban vannak a projekthez szükséges fájlok.

The angular7-app/ folder has the following **file structure** −

- **angular.json** − project name, version of cli, etc.
- **.editorconfig** − config file
- **.gitignore** −
- **package.json** − The package.json: tehát hogy milyen könyvtárakat telepítettünk a node_modules mappa milyen könyvtárakat tartalmaz. Az npm install kiadásával ezek a könyvtárak feltepülnek, (a node_modeles mappába kerülnek ha nincsenek ott).

## app

Ez több fájlt is tartalmaz, itt mindenképpen találunk egy html, css és ts fájlt.

### app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';

@NgModule({
   declarations: [
      AppComponent
   ],
   imports: [
      BrowserModule,
      AppRoutingModule
   ],
   providers: [],
   bootstrap: [AppComponent]
})
export class AppModule { }
```

@NgModule:

Declarations - A declaration-ben a komponensek referenciái tárolódnak. Az App komponens az alapértelmezett komponens, amelyet új projekt indításakor automatikusan létrehoz az angular. Új komponenseket mi magunk generálunk.

Importálás (Imports) – A modulok-at importálni kell. Jelenleg a BrowserModule az import része, amelyet @ angular / platform-browser segítségével importáltunk. Van még egy routing modul is, ez az AppRoutingModule.

Providers – Itt vannak a service-ek referenciái.

Bootstrap - Ez a létrehozandó alapértelmezett komponensre mutat, azaz az AppComponentre.

app.component.html

```
<!--The content below is only a placeholder and can be replaced.-
->
<div style = "text-align:center">
   <h1>Welcome to {{ title }}!</h1>
   <img width = "300" alt = "Angular Logo"
   src =
"
cvMjAwMC9zdmciIHZp

ZXdCb3g9IjAgMCAyNTAgMjUwIj4KICAgIDxwYXRoIGZpbGw9IiNERDAwMzEiIGQ9I
k0xMjUgMzBMMzEuOSA

2My4ybDE0LjIgMTIzLjFMMTI1IDIzMGw3OC45LTQzLjcgMTQuMi0xMjMuMXoiIC8+
CiAgICA8cGF0aCBma

WxsPSIjQzMwMDJGIiBkPSJNMTI1IDMwdjIyLjItLjFWMjMwbDc4LjktNDMuNyAxNC
4yLTEyMy4xTDEyNSA

zMHoiIC8+CiAgICA8cGF0aCZmlsbD0iI0ZGRkZGRiIgZD0iTTEyNS1Mi4xTDY2
LjggMTgyLjZoMTgyLjN2

wxMS43LTI5LjJoNDkuNGwxMS43IDI5LjJIMTgzLTEyNSA1Mi4xem0xNyA4My4zaC0
zNGwxNy00MC45IDE3I

DQwLjl6IiAvPgogIDwvc3ZnPg==">7>
</div>

<h2>Here are some links to help you start:</h2>
<ul>
   <li>
      <h2><a target = "_blank" rel = "noopener"
         href = "https://angular.io/tutorial">Tour of Heroes</a>
      </h2>
   </li>
   <li>
      <h2><a target = "_blank" rel = "noopener"
         href = https://angular.io/cli">CLI Documentation</>
      </h2>
   </li>
```

```
    <li>
       <h2><a target = "_blank" rel = "noopener"
          href = "https://blog.angular.io/">Angular blog</a>
       </h2>
    </li>
</ul>
<router-outlet></router-outlet>
```

app.component.spec.ts

Ide írjuk a tesztet.

app.component.ts

```
import { Component } from '@angular/core';
@Component({
   selector: 'app-root',
   templateUrl: './app.component.html',
   styleUrls: ['./app.component.css']
})
export class AppComponent {
   title = 'Angular 7';
}
```

app-routing.module.ts

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

const routes: Routes = [];
@NgModule({
   imports: [RouterModule.forRoot(routes)],
   exports: [RouterModule]
})
export class AppRoutingModule { }
```

Assets

Itt tárolhatjuk a képeket, js fájlokat.

Environment

Ez a mappa a production vagy a dev environment fájljait tartalmazza. A mappa két fájlt tartalmaz:

• environment.prod.ts

• environment.ts

Mindkét fájl részletezi, hogy a végleges fájlt a production vagy a dev környezetben kell-e fordítani:

Az angular7-app / mappa kiegészítő fájlszerkezete a következőket tartalmazza:

### favicon.ico

Ez egy fájl, amelyet általában egy webhely gyökérkönyvtárában talál.

### index.html

A böngésző ennek a fájlnak a tartalmát jeleníti meg.

```
<html lang = "en">
   <head>
      <meta charset = "utf-8"7gt;
      <title>Angular7App</title>
      <base href = "/">
      <meta name = "viewport" content = "width=device-width,
initial-scale=1">
      <link rel = "icon" type = "image/x-icon" href =
"favicon.ico">
   </head>
   <body>
      <app-root></app-root>
   </body>
</html>
```

A body-nak <app-root> </app-root> -ja van. Ez az a selector, amelyet az app.component.ts fájl használ, és megjeleníti az app.component.html fájl-t.

### main.ts

A main.ts az a fájl, ahonnan megkezdjük a projekt fejlesztését. A szükséges modul importálásával kezdődik. Most egy /core, angular/platform-browser-dynamic, app.module és environment vannak importálva.

```
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-
browser-dynamic';
import { AppModule } from './app/app.module';
import { environment } from './environments/environment';

if (environment.production) {
   enableProdMode();
}
platformBrowserDynamic().bootstrapModule(AppModule).catch(err =>
console.error(err));
```

A platformBrowserDynamic().bootstrapModule(AppModule) az AppModule-t adjuk meg szülő modulnak. Ezért amikor a böngészőben lefut, index.html oldalon érhető el. Az index.html belsőleg a main.ts-re utal, amely meghívja a szülő modult, azaz az AppModule-t, amikor a következő kód lefut –

```
platformBrowserDynamic().bootstrapModule(AppModule).catch(err =>
console.error(err));
```

Amikor az AppModule meghívásra kerül, akkor az app.module.ts-t hívja meg.

```
bootstrap: [AppComponent]
```

Az app.component.ts fájlban van egy selector: app-root, amelyet az index.html fájl használ. Megjeleníti az app.component.html webhelyen található tartalmat.

### polyfill.ts

Ezt a régebbi verziók kompatibilitására használják.

### styles.css

A stílusfájl.

### test.ts

Unit tesztek.

### tsconfig.app.json

Ezt a fordítás során használják, rendelkezik a konfiguráció részleteivel, amelyeket az alkalmazás futtatásához kell használni.

### tsconfig.spec.json

Tesztekhez

### typings.d.ts

A Typescript definícióra szolgál.

## Angular7 - Components

A fejlesztés nagy részét az Angular 7-rel a komponensek hajtják végre. A komponensek alapvetően olyan osztályok, amelyek kölcsönhatásba lépnek a komponensek .html fájljával. (A .html fog megjelenni a böngészőben).

Az alkalmazás komponense az alábbi fájlokból áll:

- app.component.css
- app.component.html
- app.component.spec.ts
- app.component.ts
- app.module.ts

És ha a routing-ot is kiválasztottuk a project generálása során, akkor az alábbi fájl is a komponnshez adódik:

- app-routing.module.ts

**app.module.ts**

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';

@NgModule({
   declarations: [
      AppComponent
   ],
```

```
   imports: [
      BrowserModule,
      AppRoutingModule
   ],
   providers: [],
   bootstrap: [AppComponent]
})
export class AppModule { }
```

Új komponens létrehozása:

```
ng g component new-cmp
```

# Angular7 - Modules

A modul egy olyan helyre utal, ahol össze lehet csoportosítani az alkalmazáshoz kapcsolódó dolgokat: components, directives, pipes, services.

A modul meghatározásához használhatjuk az NgModule-t. Amikor új projektet hoz létre, az ngmodule alapértelmezés szerint létrejön az app.module.ts fájlban, és a következőképpen néz ki -

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { NewCmpComponent } from './new-cmp/new-cmp.component';

@NgModule({
   declarations: [
      AppComponent,
      NewCmpComponent
   ],
   imports: [
      BrowserModule,
      AppRoutingModule
   ],
   providers: [],
   bootstrap: [AppComponent]
})
export class AppModule { }
```

Az NgModule-t be kell importálni a következőképpen:

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

Az NgModule struktúrája:

```
@NgModule({
   declarations: [
      AppComponent,
```

```
      NewCmpComponent
   ],
   imports: [
      BrowserModule,
      AppRoutingModule
   ],
   providers: [],
   bootstrap: [AppComponent]
})
```

## Declaration

Ez az létrehozott komponensek tömbje. Ha új komponens jön létre, először azt importálják, és a declaration részben felsorolják-

```
declarations: [
   AppComponent,
   NewCmpComponent
]
```

### Import

Ez egy modul tömb, amelyet az alkalmazásban használni kell . Például, most a @NgModule-ban látjuk, hogy a Browser Module-t importáltuk. Ha az alkalmazásnak form-okra van szüksége, a következő kóddal tehetjük meg-

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

```
imports: [
   BrowserModule,
   FormsModule
]
```

### Providers

Itt a service-ek vannak.

### Bootstrap

Ez a futáshoz szükséges main app komponens.

# Angular7 - Data Binding

Az adatkötéshez az alábbi operátort használjuk: {{}}.

### Example

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

@Component({
   selector: 'app-root',
   templateUrl: './app.component.html',
   styleUrls: ['./app.component.css']
})
```

```
export class AppComponent {
   title = 'Angular 7';

   // declared array of months.
   months = ["January", "February", "March", "April", "May",
"June", "July",
      "August", "September", "October", "November", "December"];

   isavailable = true; //variable is set to true
}
```

**app.component.html**

```
<!--The content below is only a placeholder and can be replaced.-
->
<div style = "text-align:center">
   <h1> Welcome to {{title}}. </h1>
</div>

<div> Months :
   <select>
      <option *ngFor = "let i of months">{{i}}</option>
   </select>
</div>
<br/>

<div>
   <span *ngIf = "isavailable">Condition is valid.</span>
   //over here based on if condition the text condition is valid
is displayed.
   //If the value of isavailable is set to false it will not
display the text.
</div>
```

## Example

**if else**

```
import { Component } from '@angular/core';
@Component({
   selector: 'app-root',
   templateUrl: './app.component.html',
   styleUrls: ['./app.component.css']
})
export class AppComponent {
   title = 'Angular 7';

   // declared array of months.
   months = ["January", "Feburary", "March", "April",
"May","June", "July",
      "August", "September", "October", "November", "December"];

   isavailable = false; //variable is set to true
```

```
}
```

```
<!--The content below is only a placeholder and can be replaced.-
->
<div style = "text-align:center">
   <h1> Welcome to {{title}}. </h1>
</div>

<div> Months :
   <select>
      <option *ngFor = "let i of months">{{i}}</option>
   </select>
</div>
<br/>

<div>
   <span *ngIf = "isavailable; else condition1">Condition is
valid.</span>
   <ng-template #condition1>Condition is invalid</ng-template>
</div>
```

**if then else**

```
import { Component } from '@angular/core';
@Component({
   selector: 'app-root',
   templateUrl: './app.component.html',
   styleUrls: ['./app.component.css']
})
export class AppComponent {
   title = 'Angular 7';

   // declared array of months.
   months = ["January", "February", "March", "April", "May",
"June", "July",
      "August", "September", "October", "November", "December"];

   isavailable = true; //variable is set to true
}
```

```
<!--The content below is only a placeholder and can be replaced.-
->
<div style = "text-align:center">
   <h1> Welcome to {{title}}. </h1>
</div>

<div> Months :
   <select>
      <option *ngFor="let i of months">{{i}}</option>
   </select>
```

```
</div>
<br/>

<div>
   <span *ngIf = "isavailable; then condition1 else condition2">
      Condition is valid.
   </span>
   <ng-template #condition1>Condition is valid</ng-template>
   <ng-template #condition2>Condition is invalid</ng-template>
</div>
```

## Angular7 - Event Binding

***app.component.html***

```
<!--The content below is only a placeholder and can be replaced.-
->
<div style = "text-align:center">
   <h1>Welcome to {{title}}.</h1>
</div>

<div> Months :
   <select>
      <option *ngFor = "let i of months">{{i}}</option>
   </select>
</div>
<br/>

<div>
   <span *ngIf = "isavailable; then condition1 else condition2">
      Condition is valid.
   </span>
   <ng-template #condition1>Condition is valid</ng-template>
   <ng-template #condition2>Condition is invalid</ng-template>
</div>
<button (click) = "myClickFunction($event)">
   Click Me
</button>
```

**app.component.ts**

```
import { Component } from '@angular/core';
@Component({
   selector: 'app-root',
   templateUrl: './app.component.html',
   styleUrls: ['./app.component.css']
})
export class AppComponent {
   title = 'Angular 7';

   // declared array of months.
```

```
   months = ["January", "February", "March", "April",
"May","June", "July",
      "August", "September", "October", "November", "December"];

   isavailable = true; //variable is set to true
   myClickFunction(event) {
      //just added console.log which will display the event
details in browser on click of the button.
      alert("Button is clicked");
      console.log(event);
   }
}
```

add.component.css –

```
button {
   background-color: #2B3BCF;
   border: none;
   color: white;
   padding: 10px 10px;
   text-align: center;
   text-decoration: none;
   display: inline-block;
   font-size: 20px;
}
```

onchange event to the dropdown

***app.component.html***

```
<!--The content below is only a placeholder and can be replaced.-
->
<div style = "text-align:center">
   <h1>Welcome to {{title}}.</h1>
</div>

<div> Months :
   <select (change) = "changemonths($event)">
      <option *ngFor = "let i of months">{{i}}</option>
   </select>
</div>
<br/>

<div>
   <span *ngIf = "isavailable; then condition1 else condition2">
      Condition is valid.
   </span>
   <ng-template #condition1>Condition is valid</ng-template>
   <ng-template #condition2>Condition is invalid</ng-template>
</div>
<br/>
```

```
<button (click) = "myClickFunction($event)">
   Click Me
</button>
```

**app.component.ts** file –

```
import { Component } from '@angular/core';
@Component({
   selector: 'app-root',
   templateUrl: './app.component.html',
   styleUrls: ['./app.component.css']
})
export class AppComponent {
   title = 'Angular 7';

   // declared array of months.
   months = ["January", "Feburary", "March", "April", "May",
"June", "July",
      "August", "September", "October", "November", "December"];

   isavailable = true; //variable is set to true
   myClickFunction(event) {
      //just added console.log which will display the event
      details in browser on click of the button.
      alert("Button is clicked");
      console.log(event);
   }
   changemonths(event) {
      console.log("Changed month from the Dropdown");
      console.log(event);
   }
}
```

## Angular7 - Templates

***app.component.html***

```
<!--The content below is only a placeholder and can be replaced.-
->
<div style = "text-align:center">
   <h1>Welcome to {{title}}.</h1>
</div>

<div> Months :
   <select (change) = "changemonths($event)" name = "month">
      <option *ngFor = "let i of months">{{i}}</option>
   </select>
</div>
<br/>

<div>
```

```
   <span *ngIf = "isavailable;then condition1 else condition2">
      Condition is valid.
   </span>
   <ng-template #condition1>Condition is valid from template</ng-
template>
   <ng-template #condition2>Condition is invalid from
template</ng-template>
</div>
<button (click) = "myClickFunction($event)">Click Me</button>
```

***app.component.ts***

```
import { Component } from '@angular/core';
@Component({
   selector: 'app-root',
   templateUrl: './app.component.html',
   styleUrls: ['./app.component.css']
})
export class AppComponent {
   title = 'Angular 7';

   // declared array of months.
   months = ["January", "February", "March", "April", "May",
"June", "July",
      "August", "September", "October", "November", "December"];
   isavailable = false; // variable is set to true

   myClickFunction(event) {
      //just added console.log which will display the event
details in browser on click of the button.
      alert("Button is clicked");
      console.log(event);
   }
   changemonths(event) {
      alert("Changed month from the Dropdown");
   }
}
```

```
<!--The content below is only a placeholder and can be replaced.-
->
<div style = "text-align:center">
   <h1> Welcome to {{title}}. </h1>
</div>

<div> Months :
   <select (change) = "changemonths($event)" name = "month">
      <option *ngFor = "let i of months">{{i}}</option>
   </select>
</div>
<br/>
```

```
<div>
   <span *ngIf = "isavailable; else condition2">
      Condition is valid.
   </span>
   <ng-template #condition1>Condition is valid from template
</ng-template>
   <ng-template #condition2>Condition is invalid from
template</ng-template>
</div>
<button (click) = "myClickFunction($event)">Click Me</button>
```

# Angular7 - Directives

A direktívák javascript osztályok, amelyeket a @directive segítségével deklarálunk. Az Angular-ban 3 direktíva van:

### Component Directives

Meghatározza a komponensek feldolgozását, példányosítását és futásidejű használatát.

### Structural Directives

A strukturális direktívák alapvetően a dom elemek manipulálásával foglalkoznak. A strukturális direktíváknak * jelük van a direktíva előtt. Például: * nglf és * ngFor.

### Attribute Directives

Az attribútum direktívák a dom elem megjelenésének és viselkedésének megváltoztatásával foglalkoznak. Saját direktívát is készíthetünk, ezt a következőkben nézzük meg:

## How to Create Custom Directives?

Ezeket a direktívákat mi magunk készítjük.

***app.module.ts***

```
import { BrowserModule } from'@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { NewCmpComponent } from'./new-cmp/new-cmp.component';
import { ChangeTextDirective } from './change-text.directive';

@NgModule({
   declarations: [
      AppComponent,
      NewCmpComponent,
      ChangeTextDirective
   ],
   imports: [
      BrowserModule,
```

```
      AppRoutingModule
   ],
   providers: [],
   bootstrap: [AppComponent]
})
export class AppModule { }
```

app.component.html

```
<!--The content below is only a placeholder and can be replaced.-
->
<div style = "text-align:center">
   <h1> Welcome to {{title}}. </h1>
</div>
<div style = "text-align:center">
   <span changeText >Welcome to {{title}}.</span>
</div>
```

***change-text.directive.ts***

```
import { Directive, ElementRef} from '@angular/core';
@Directive({
   selector: '[changeText]'
})
export class ChangeTextDirective {
   constructor(Element: ElementRef) {
      console.log(Element);
      Element.nativeElement.innerText = "Text is changed by
changeText Directive.";
   }
}
```

## Angular7 - Pipes

Az adat átalakítására a | karaktert használjuk.

```
{{ Welcome to Angular 7 | lowercase}}
```

Egész számok, karakterláncok, tömbök és dátum során a | konvertál a kívánt formátumban.

***app.component.ts***

```
import { Component } from '@angular/core';
@Component({
   selector: 'app-root',
   templateUrl: './app.component.html',
   styleUrls: ['./app.component.css']
})
export class AppComponent {
   title = 'Angular 7 Project!';
}
```

**app.component.html** file −

```
<b>{{title | uppercase}}</b><br/>
<b>{{title | lowercase}}</b>
```

Néhány beépített pipe:

- Lowercasepipe
- Uppercasepipe
- Datepipe
- Currencypipe
- Jsonpipe
- Percentpipe
- Decimalpipe
- Slicepipe

**app.component.ts** file −

```
import { Component } from '@angular/core';
@Component({
   selector: 'app-root',
   templateUrl: './app.component.html',
   styleUrls: ['./app.component.css']
})
export class AppComponent {
   title = 'Angular 7 Project!';
   todaydate = new Date();
   jsonval = {name:'Rox', age:'25', address:{a1:'Mumbai',
a2:'Karnataka'}};
   months = ["Jan", "Feb", "Mar", "April", "May", "Jun", "July",
"Aug",
      "Sept", "Oct", "Nov", "Dec"];
}
```

**app.component.html** file as shown below −

```
<!--The content below is only a placeholder and can be replaced.-
->
<div style = "width:100%;">
   <div style = "width:40%;float:left;border:solid 1px black;">
      <h1>Uppercase Pipe</h1>
      <b>{{title | uppercase}}</b>
      <br/>

      <h1>Lowercase Pipe</h1>
      <b>{{title | lowercase}}</b>
      <h1>Currency Pipe</h1>
      <b>{{6589.23 | currency:"USD"}}</b>
```

```
      <br/>

      <b>{{6589.23 | currency:"USD":true}}</b>
      // Boolean true is used to get the sign of the currency.
      <h1>Date pipe</h1>
      <b>{{todaydate | date:'d/M/y'}}</b>
      <br/>

      <b>{{todaydate | date:'shortTime'}}</b>
      <h1>Decimal Pipe</h1>
      <b>{{ 454.78787814 | number: '3.4-4' }}</b>
      // 3 is for main integer, 4 -4 are for integers to be
displayed.
   </div>

   <div style = "width:40%;float:left;border:solid 1px black;"<
      <h1<Json Pipe</h1>
      <b>{{ jsonval | json }}</b>
      <h1>Percent Pipe</h1>
      <b>{{00.54565 | percent}}</b>
      <h1>Slice Pipe</h1>
      <b>{{months | slice:2:6}}</b>
      // here 2 and 6 refers to the start and the end index
   </div>
</div>
```

## How to Create a Custom Pipe?

***app.sqrt.ts***

```
import {Pipe, PipeTransform} from '@angular/core';
@Pipe ({
   name : 'sqrt'
})
export class SqrtPipe implements PipeTransform {
   transform(val : number) : number {
      return Math.sqrt(val);
   }
}
```

**app.module.ts**. This is done as follows −

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { NewCmpComponent } from './new-cmp/new-cmp.component';
import { ChangeTextDirective } from './change-text.directive';
import { SqrtPipe } from './app.sqrt';
```

```
@NgModule({
   declarations: [
      SqrtPipe,
      AppComponent,
      NewCmpComponent,
      ChangeTextDirective
   ],
   imports: [
      BrowserModule,
      AppRoutingModule
   ],
   providers: [],
   bootstrap: [AppComponent]
})
export class AppModule { }
```

**app.component.html** file.

```
<h1>Custom Pipe</h1>
<b>Square root of 25 is: {{25 | sqrt}}</b>
<br/>
<b>Square root of 729 is: {{729 | sqrt}}</b>
```

## Angular7 - Routing

**app.module.ts**

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { NewCmpComponent } from './new-cmp/new-cmp.component';
import { ChangeTextDirective } from './change-text.directive';
import { SqrtPipe } from './app.sqrt';

@NgModule({
   declarations: [
      SqrtPipe,
      AppComponent,
      NewCmpComponent,
      ChangeTextDirective
   ],
   imports: [
      BrowserModule,
      AppRoutingModule
   ],
   providers: [],
   bootstrap: [AppComponent]
})
export class AppModule { }
```

**app-routing.module** fájl

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

const routes: Routes = [];
@NgModule({
   imports: [
      RouterModule.forRoot(routes)
   ],
   exports: [RouterModule]
})
export class AppRoutingModule { }
```

app.component.html fájl

```
<h1>Angular 7 Routing Demo</h1>
<router-outlet></router-outlet>
```

## Component Home

```
ng g component home
```

### Component Contact Us

```
ng g component contactus
```

app.module.ts –

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { NewCmpComponent } from './new-cmp/new-cmp.component';
import { ChangeTextDirective } from './change-text.directive';
import { SqrtPipe } from './app.sqrt';
import { HomeComponent } from './home/home.component';
import { ContactusComponent } from
'./contactus/contactus.component';

@NgModule({
   declarations: [
      SqrtPipe,
      AppComponent,
      NewCmpComponent,
      ChangeTextDirective,
      HomeComponent,
      ContactusComponent
   ],
   imports: [
```

```
      BrowserModule,
      AppRoutingModule
   ],
   providers: [],
   bootstrap: [AppComponent]
})
export class AppModule { }
```

**app-routing.module**.ts as shown below −

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { HomeComponent } from './home/home.component';
import { ContactusComponent } from
'./contactus/contactus.component';

const routes: Routes = [
   {path:"home", component:HomeComponent},
   {path:"contactus", component:ContactusComponent}
];
@NgModule({
   imports: [RouterModule.forRoot(routes)],
   exports: [RouterModule]
})
export class AppRoutingModule { }
```

app.module.ts as follows −

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppRoutingModule , RoutingComponent} from './app-
routing.module';
import { AppComponent } from './app.component';
import { NewCmpComponent } from './new-cmp/new-cmp.component';
import { ChangeTextDirective } from './change-text.directive';
import { SqrtPipe } from './app.sqrt';

@NgModule({
   declarations: [
      SqrtPipe,
      AppComponent,
      NewCmpComponent,
      ChangeTextDirective,
      RoutingComponent
   ],
   imports: [
      BrowserModule,
      AppRoutingModule
   ],
   providers: [],
   bootstrap: [AppComponent]
```

```
})
export class AppModule { }
```

**app.component.html**

```
<h1>Angular 7 Routing Demo</h1>
<nav>
   <a routerLink = "/home">Home</a>
   <a routerLink = "/contactus">Contact Us </a>
</nav>
<router-outlet></router-outlet>
```

app.component.css −

```
a:link, a:visited {
   background-color: #848686;
   color: white;
   padding: 10px 25px;
   text-align: center;
   text-decoration: none;
   display: inline-block;
}
a:hover, a:active {
   background-color: #BD9696;
}
```

## Angular7 - Services

Lehetséges, hogy valamilyen kódra szükségünk van mindenhol. Például az adatot kell megosztani a komponensek között. Ezt a service segítségével érjük el. A service-ekkel az egész projekten belül elérhetünk properti-ket

```
ng g service myservice
```

**myservice.service.ts**

```
import { Injectable } from '@angular/core';
@Injectable({
   providedIn: 'root'
})
export class MyserviceService {
   constructor() { }
}
```

Mielőtt elkészítenénk az új servie-t include-olnunk kell a main-ben

**app.module.ts**.

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

```
import { AppRoutingModule , RoutingComponent} from './app-
routing.module';
import { AppComponent } from './app.component';
import { NewCmpComponent } from './new-cmp/new-cmp.component';
import { ChangeTextDirective } from './change-text.directive';
import { SqrtPipe } from './app.sqrt';
import { MyserviceService } from './myservice.service';

@NgModule({
   declarations: [
      SqrtPipe,
      AppComponent,
      NewCmpComponent,
      ChangeTextDirective,
      RoutingComponent
   ],
   imports: [
      BrowserModule,
      AppRoutingModule
   ],
   providers: [MyserviceService],
   bootstrap: [AppComponent]
})
export class AppModule { }
```

```
import { Injectable } from '@angular/core';
@Injectable({
   providedIn: 'root'
})
export class MyserviceService {
   constructor() { }
   showTodayDate() {
      let ndate = new Date();
      return ndate;
   }
}
```

**app.component.ts**

```
import { Component } from '@angular/core';
import { MyserviceService } from './myservice.service';
@Component({ selector: 'app-root',
   templateUrl: './app.component.html',
   styleUrls: ['./app.component.css']
})
export class AppComponent {
   title = 'Angular 7 Project!';
   todaydate;
   constructor(private myservice: MyserviceService) {}
   ngOnInit() {
      this.todaydate = this.myservice.showTodayDate();
```

```
    }
}
```

**app.component.html**

```
{{todaydate}}
<app-new-cmp></app-new-cmp>
```

**new-cmp.component.ts**

```
import { Component, OnInit } from '@angular/core';
import { MyserviceService } from './../myservice.service';

@Component({
   selector: 'app-new-cmp',
   templateUrl: './new-cmp.component.html',
   styleUrls: ['./new-cmp.component.css']
})
export class NewCmpComponent implements OnInit {
   newcomponent = "Entered in new component created";
   todaydate;
   constructor(private myservice: MyserviceService) { }
   ngOnInit() {
      this.todaydate = this.myservice.showTodayDate();
   }
}
```

**new-cmp.component.html**

```
<p>
   {{newcomponent}}
</p>
<p>
   Today's Date : {{todaydate}}
</p>
```

**myservice.service.ts**

```
import { Injectable } from '@angular/core';
@Injectable({
   providedIn: 'root'
})
export class MyserviceService {
   serviceproperty = "Service Created";
   constructor() { }
   showTodayDate() {
      let ndate = new Date();
      return ndate;
   }
}
```

**app.component.ts**,

```
import { Component } from '@angular/core';
import { MyserviceService } from './myservice.service';
@Component({
   selector: 'app-root',
   templateUrl: './app.component.html',
   styleUrls: ['./app.component.css']
})
export class AppComponent {
   title = 'Angular 7 Project!';
   todaydate;
   componentproperty;
   constructor(private myservice: MyserviceService) {}
   ngOnInit() {
      this.todaydate = this.myservice.showTodayDate();
      console.log(this.myservice.serviceproperty);
      this.myservice.serviceproperty = "component created";
      // value is changed. this.componentproperty =
      this.myservice.serviceproperty;
   }
}
```

```
import { Component, OnInit } from '@angular/core';
import { MyserviceService } from './../myservice.service';
@Component({
   selector: 'app-new-cmp',
   templateUrl: './new-cmp.component.html',
   styleUrls: ['./new-cmp.component.css']
})
export class NewCmpComponent implements OnInit {
   todaydate;
   newcomponentproperty; newcomponent = "Entered in
   newcomponent"; constructor(private myservice:
   MyserviceService) {}
   ngOnInit() {
      this.todaydate = this.myservice.showTodayDate();
      this.newcomponentproperty =
      this.myservice.serviceproperty;
   }
}
```

**app.component.html**

```
<h3>{{todaydate}}>/h3>
<h3> Service Property : {{componentproperty}} </h3>
<app-new-cmp></app-new-cmp>
```

**new-cmp.component.html**

```
<h3>{{newcomponent}} </h3>
<h3> Service Property : {{newcomponentproperty}} </h3>
```

```
<h3> Today's Date : {{todaydate}} </h3>
```

## Angular7 - Http Client

A HttpClient segít külsõ adatok beolvasásában, elküldésében stb. A http szolgáltatás használatához importálnunk kell a http modult. A http-szolgáltatás használatának megkezdéséhez importálnunk kell a modult az app.module.ts-ben, az alább látható módon - a HttpClientModule-ot importáljuk a @ angular / common / http-ből, és ezt az import tömbben is jelezzük.

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppRoutingModule , RoutingComponent} from './app-
routing.module';
import { AppComponent } from './app.component';
import { NewCmpComponent } from './new-cmp/new-cmp.component';
import { ChangeTextDirective } from './change-text.directive';
import { SqrtPipe } from './app.sqrt';
import { MyserviceService } from './myservice.service';
import { HttpClientModule } from '@angular/common/http';

@NgModule({
   declarations: [
      SqrtPipe,
      AppComponent,
      NewCmpComponent,
      ChangeTextDirective,
      RoutingComponent
   ],
   imports: [
      BrowserModule,
      AppRoutingModule,
      HttpClientModule
   ],
   providers: [MyserviceService],
   bootstrap: [AppComponent]
})
export class AppModule { }
```

Az adatokat a kiszolgálótól a fent említett httpclient modul segítségével kérdezzük le. Ezt az előző fejezetben létrehozott service-en belül fogjuk megtenni, és az általunk kívánt komponenseken belül fogjuk az adatokat felhasználni.

**myservice.service.ts**

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
@Injectable({
   providedIn: 'root'
})
export class MyserviceService {
   private finaldata = [];
   private apiurl = "http://jsonplaceholder.typicode.com/users";
```

```
   constructor(private http: HttpClient) { }
   getData() {
      return this.http.get(this.apiurl);
   }
}
```

A getData metódus a lekért adattal tér vissza. A getData metódust az app.component.ts-ből hívjuk meg:

```
import { Component } from '@angular/core';
import { MyserviceService } from './myservice.service';
@Component({
   selector: 'app-root',
   templateUrl: './app.component.html',
   styleUrls: ['./app.component.css']
})
export class AppComponent {
   title = 'Angular 7 Project!';
   public persondata = [];
   constructor(private myservice: MyserviceService) {}
   ngOnInit() {
      this.myservice.getData().subscribe((data) => {
         this.persondata = Array.from(Object.keys(data),
k=>data[k]);
         console.log(this.persondata);
      });
   }
}
```

app.component.html as follows −

```
<h3>Users Data</h3>
<ul>
   <li *ngFor="let item of persondata; let i = index"<
      {{item.name}}
   </li>
</ul>
```

## Angular7 - CLI Prompts

install:

```
npm install -g @angular/cli
```

Új projekt létrehozása

```
ng new PROJECT-NAME
cd PROJECT-NAME
ng serve
```

```
http://localhost:4200/
```

4200 az alapértelmezett port, ha ezt meg szeretnénk változtatni, akkor az alábbi paranccsal tehetjük meg:

```
ng serve --host 0.0.0.0 --port 4201
```

## Command for Angular Update

```
ng update @angular/cli @angular/core
```

## Angular Important Command List

| Sr.No | Commands and Description |
|---|---|
| 1 | **Component**<br>ng g component new-component |
| 2 | **Directive**<br>ng g directive new-directive |
| 3 | **Pipe**<br>ng g pipe new-pipe |
| 4 | **Service**<br>ng g service new-service |
| 5 | **Module**<br>ng g module my-module |
| 6 | **Test**<br>ng test |
| 7 | **Build**<br>ng build --configuration=production // for production environment<br>ng build --configuration=staging // for stating environment |

# Angular7 - Forms

Két féleképpen kezelhetjük a form-okat:

- Template driven form
- Model driven form

## Template Driven Form

Egy template driven form során a munka nagy része template-ben történik. A model driven form során a munka nagy része az component osztályban történik.

Először a template driven formot nézzük meg. Készítünk egy egyszerű bejelentkezési űrlapot, és hozzáadjuk az e-mail azonosítót, a jelszót és a beküldés gombot az űrlaphoz. Először be kell importálnunk a FormsModule-ba a @ angular / form fájlokat, amelyet az app.module.ts-ben hajtunk végre az alábbiak szerint –

**app.module.ts**

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppRoutingModule , RoutingComponent} from './app-
routing.module';
import { AppComponent } from './app.component';
import { NewCmpComponent } from './new-cmp/new-cmp.component';
import { ChangeTextDirective } from './change-text.directive';
import { SqrtPipe } from './app.sqrt';
import { MyserviceService } from './myservice.service';
import { HttpClientModule } from '@angular/common/http';
import { ScrollDispatchModule } from '@angular/cdk/scrolling';
import { DragDropModule } from '@angular/cdk/drag-drop';
import { FormsModule } from '@angular/forms';

@NgModule({
   declarations: [
      SqrtPipe,
      AppComponent,
      NewCmpComponent,
      ChangeTextDirective,
      RoutingComponent
   ],
   imports: [
      BrowserModule,
      AppRoutingModule,
      HttpClientModule,
      ScrollDispatchModule,
      DragDropModule,
      FormsModule
   ],
   providers: [MyserviceService],
   bootstrap: [AppComponent]
})
export class AppModule { }
```

**app.component.html** file.

```
<form #userlogin = "ngForm" (ngSubmit) =
"onClickSubmit(userlogin.value)">
   <input type = "text" name = "emailid" placeholder = "emailid"
ngModel>
```

```
   <br/>
   <input type = "password" name = "passwd" placeholder =
"passwd" ngModel>
   <br/>
   <input type = "submit" value = "submit">
</form>
```

A template driven form során létre kell hoznunk a modell-vezérlőket az ngModel direktíva és a name attribútum hozzáadásával.

Ahogy látható, az ngForm-ot a #userlogin-hez is hozzáadtuk. Az ngForm direktívát a létrehozott form template-hez is hozzá kell adni. Az onClickSubmit függvényt is hozzáadtuk a userlogin.value-hoz.

Let us now create the function in the **app.component.ts**

```
import { Component } from '@angular/core';
import { MyserviceService } from './myservice.service';

@Component({
   selector: 'app-root',
   templateUrl: './app.component.html',
   styleUrls: ['./app.component.css']
})
export class AppComponent {
   title = 'Angular 7 Project!';
   constructor(private myservice: MyserviceService) { }
   ngOnInit() { }
   onClickSubmit(data) {
      alert("Entered Email id : " + data.emailid);
   }
}
```

**app.component.css –**

```
input[type = text], input[type = password] {
   width: 40%;
   padding: 12px 20px;
   margin: 8px 0;
   display: inline-block;
   border: 1px solid #B3A9A9;
   box-sizing: border-box;
}
input[type = submit] {
   padding: 12px 20px;
   margin: 8px 0;
   display: inline-block;
   border: 1px solid #B3A9A9;
   box-sizing: border-box;
}
```

## Model Driven Form

A model driven form során be kell importálnunk a ReactiveFormsModule-t, és az import tömbben is jelezni kell.

**app.module.ts**.

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppRoutingModule , RoutingComponent} from './app-
routing.module';
import { AppComponent } from './app.component';
import { NewCmpComponent } from './new-cmp/new-cmp.component';
import { ChangeTextDirective } from './change-text.directive';
import { SqrtPipe } from './app.sqrt';
import { MyserviceService } from './myservice.service';
import { HttpClientModule } from '@angular/common/http';
import { ScrollDispatchModule } from '@angular/cdk/scrolling';
import { DragDropModule } from '@angular/cdk/drag-drop';
import { ReactiveFormsModule } from '@angular/forms';

@NgModule({
   declarations: [
      SqrtPipe,
      AppComponent,
      NewCmpComponent,
      ChangeTextDirective,
      RoutingComponent
   ],
   imports: [
      BrowserModule,
      AppRoutingModule,
      HttpClientModule,
      ScrollDispatchModule,
      DragDropModule,
      ReactiveFormsModule
   ],
   providers: [MyserviceService],
   bootstrap: [AppComponent]
})
export class AppModule { }
```

Az **app.component.ts** –ben kell importálni az egyes model driven form modulokat, például itt a **FormGroup, FormControl** –t importáltuk.

```
import { Component } from '@angular/core';
import { MyserviceService } from './myservice.service';
import { FormGroup, FormControl } from '@angular/forms';

@Component({
   selector: 'app-root',
   templateUrl: './app.component.html',
   styleUrls: ['./app.component.css']
```

```
})
export class AppComponent {
   title = 'Angular 7 Project!';
   emailid;
   formdata;
   constructor(private myservice: MyserviceService) { }
   ngOnInit() {
      this.formdata = new FormGroup({
         emailid: new FormControl("angular@gmail.com"),
         passwd: new FormControl("abcd1234")
      });
   }
   onClickSubmit(data) {this.emailid = data.emailid;}
}
```

**app.component.html**.

```
<div>
   <form [formGroup] = "formdata" (ngSubmit) =
"onClickSubmit(formdata.value)" >
      <input type = "text" class = "fortextbox" name = "emailid"
placeholder = "emailid"
         formControlName = "emailid">
      <br/>

      <input type = "password" class = "fortextbox" name =
"passwd"
         placeholder = "passwd" formControlName = "passwd">
      <br/>

      <input type = "submit" class = "forsubmit" value = "Log
In">
   </form>
</div>
<p> Email entered is : {{emailid}} </p>
```

## Form Validation

Form validáció model driven form-al. A beépített form validációt is használhatjuk. A Validators-t kell importálni a @angular/forms-ból.

```
import { FormGroup, FormControl, Validators} from
'@angular/forms'
```

Az Angular beépített validátorokkal rendelkezik, mint például **mandatory field**, **minlength**, **maxlength, pattern.** Ezeket a Validators modul segítségével lehet elérni.

app.component.ts.

```
import { Component } from '@angular/core';
import { FormGroup, FormControl, Validators} from
'@angular/forms';

@Component({
   selector: 'app-root',
   templateUrl: './app.component.html',
   styleUrls: ['./app.component.css']
})
export class AppComponent {
   title = 'Angular 4 Project!';
   todaydate;
   componentproperty;
   emailid;
   formdata;
   ngOnInit() {
      this.formdata = new FormGroup({
         emailid: new FormControl("", Validators.compose([
            Validators.required,
            Validators.pattern("[^ @]*@[^ @]*")
         ])),
         passwd: new FormControl("")
      });
   }
   onClickSubmit(data) {this.emailid = data.emailid;}
}
```

A fenti példában a required és a pattern validator-okat használtuk.

Az app.component.html-ben a submit gomb disabled, ha a form input-ok helytelenek.

```
<div>
   <form [formGroup] = "formdata" (ngSubmit) =
"onClickSubmit(formdata.value)">
      <input type = "text" class = "fortextbox" name = "emailid"
         placeholder = "emailid" formControlName = "emailid">
      <br/>

      <input type = "password" class = "fortextbox" name =
"passwd"
         placeholder = "passwd" formControlName = "passwd">
      <br/>

      <input type = "submit" [disabled] = "!formdata.valid" class
= "forsubmit"
         value = "Log In">
   </form>
</div>
<p> Email entered is : {{emailid}} </p>
```

Saját validator-hoz egy függvényt kell írni.

```
import { Component } from '@angular/core';
import { FormGroup, FormControl, Validators} from
'@angular/forms';

@Component({
   selector: 'app-root',
   templateUrl: './app.component.html',
   styleUrls: ['./app.component.css']
})
export class AppComponent {
   title = 'Angular 7 Project!';
   todaydate;
   componentproperty;
   emailid;
   formdata;
   ngOnInit() {
      this.formdata = new FormGroup({
         emailid: new FormControl("", Validators.compose([
            Validators.required,
            Validators.pattern("[^ @]*@[^ @]*")
         ])),
         passwd: new FormControl("", this.passwordvalidation)
      });
   }
   passwordvalidation(formcontrol) {
      if (formcontrol.value.length < 5) {
         return {"passwd" : true};
      }
   }
   onClickSubmit(data) {this.emailid = data.emailid;}
}
```

# Angular7 – Materials

```
npm install @angular/cdk -save
```

```
npm install --save @angular/material
```

**app.module.ts**

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppRoutingModule , RoutingComponent} from './app-
routing.module';
import { AppComponent } from './app.component';
import { NewCmpComponent } from './new-cmp/new-cmp.component';
import { ChangeTextDirective } from './change-text.directive';
import { SqrtPipe } from './app.sqrt';
```

```
import { MyserviceService } from './myservice.service';
import { HttpClientModule } from '@angular/common/http';
import { ScrollDispatchModule } from '@angular/cdk/scrolling';
import { DragDropModule } from '@angular/cdk/drag-drop';
import { ReactiveFormsModule } from '@angular/forms';
import { BrowserAnimationsModule } from '@angular/platform-
browser/animations';
import { MatButtonModule } from '@angular/material/button';
import { MatMenuModule } from '@angular/material/menu';
@NgModule({
   declarations: [
      SqrtPipe,
      AppComponent,
      NewCmpComponent,
      ChangeTextDirective,
      RoutingComponent
   ],
   imports: [
      BrowserModule,
      AppRoutingModule,
      HttpClientModule,
      ScrollDispatchModule,
      DragDropModule,
      ReactiveFormsModule,
      BrowserAnimationsModule,
      MatButtonModule,
      MatMenuModule
   ],
   providers: [MyserviceService],
   bootstrap: [AppComponent]
})
export class AppModule { }
```

A material importok az **@angular/materials** –ból kell az egyes modulokat importálni a material-hoz.

```
import { MatButtonModule } from '@angular/material/button';

import { MatMenuModule } from '@angular/material/menu';
```

**app.component.ts**

```
import { Component } from '@angular/core';
@Component({
   selector: 'app-root',
   templateUrl: './app.component.html',
   styleUrls: ['./app.component.css']
})
export class AppComponent {
   constructor() {}
}
```

A material-css support a **styles.css**-ben.

```
@import "~@angular/material/prebuilt-themes/indigo-pink.css";
```

**app.component.html**

## Menu

```
<button mat-button [matMenuTriggerFor] = "menu">Menu</button>
<mat-menu #menu = "matMenu">
   <button mat-menu-item> File </button>
   <button mat-menu-item> Save As </button>
</mat-menu>
```

## Datepicker

**app.module.ts**

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppRoutingModule , RoutingComponent} from './app-
routing.module';
import { AppComponent } from './app.component';
import { NewCmpComponent } from './new-cmp/new-cmp.component';
import { ChangeTextDirective } from './change-text.directive';
import { SqrtPipe } from './app.sqrt';
import { MyserviceService } from './myservice.service';
import { HttpClientModule } from '@angular/common/http';
import { ScrollDispatchModule } from '@angular/cdk/scrolling';
import { DragDropModule } from '@angular/cdk/drag-drop';
import { ReactiveFormsModule } from '@angular/forms';
import { BrowserAnimationsModule } from '@angular/platform-
browser/animations';
import { MatDatepickerModule} from
'@angular/material/datepicker';
import { MatInputModule} from '@angular/material/input';
import {MatNativeDateModule} from '@angular/material/core';
@NgModule({
   declarations: [
      SqrtPipe,
      AppComponent,
      NewCmpComponent,
      ChangeTextDirective,
      RoutingComponent
   ],
   imports: [
      BrowserModule,
      AppRoutingModule,
      HttpClientModule,
      ScrollDispatchModule,
      DragDropModule,
      ReactiveFormsModule,
      BrowserAnimationsModule,
      MatDatepickerModule,
```

```
      MatInputModule,
      MatNativeDateModule
   ],
   providers: [MyserviceService],
   bootstrap: [AppComponent]
})
export class AppModule { }
```

**app.component.ts**

```
import { Component } from '@angular/core';
@Component({
   selector: 'app-root',
   templateUrl: './app.component.html', s
   tyleUrls: ['./app.component.css']
})
export class AppComponent {
   constructor() {}
}
```

**app.component.html**

```
<mat-form-field>
   <input matInput [matDatepicker] = "picker" placeholder =
"Choose a date">
   <mat-datepicker-toggle matSuffix [for] = "picker"></mat-
datepicker-toggle>
   <mat-datepicker #picker></mat-datepicker>
</mat-form-field>
```

**style.css**

```
/* You can add global styles to this file, and also
import other style files */
@import '~@angular/material/prebuilt-themes/deeppurple-
amber.css';
body {
   font-family: Roboto, Arial, sans-serif;
   margin: 10;
}
.basic-container {
   padding: 30px;
}
.version-info {
   font-size: 8pt;
   float: right;
}
```

## Testing Angular 7 Project

A projekt beállítása során a teszteléshez szükséges csomagok már telepítve vannak. Minden componenshez, service-hez, direktívához készül egy .spec.ts fájl. A jasmine-t fogjuk használni a teszt esetek írására.

Az componenshez, service-hez, direktívához vagy bármilyen más fájlhoz a teszt eseteit a megfelelő .spec.ts fájlokba írjuk.

A teszt esetek futtatásához a következő parancs használható

```
ng test
```

**app.component.spec.ts –**

```
import { TestBed, async } from '@angular/core/testing';
import { RouterTestingModule } from '@angular/router/testing';
import { AppComponent } from './app.component';

describe('AppComponent', () => {
   beforeEach(async(() => {
      TestBed.configureTestingModule({
         imports: [
            RouterTestingModule
         ],
         declarations: [
            AppComponent
         ],
      }).compileComponents();
   }));
   it('should create the app', () => {
      const fixture = TestBed.createComponent(AppComponent);
      const app = fixture.debugElement.componentInstance;
      expect(app).toBeTruthy();
   });
   it(`should have as title 'angular7-app'`, () => {
      const fixture = TestBed.createComponent(AppComponent);
      const app = fixture.debugElement.componentInstance;
      expect(app.title).toEqual('angular7-app');
   });
   it('should render title in a h1 tag', () => {
      const fixture = TestBed.createComponent(AppComponent);
      fixture.detectChanges();
      const compiled = fixture.debugElement.nativeElement;
      expect(compiled.querySelector('h1').textContent).toContain(
         'Welcome to angular7-app!');
   })
});
```

**app.component.ts**

```
import { Component } from '@angular/core';
@Component({
   selector: 'app-root',
   templateUrl: './app.component.html',
   styleUrls: ['./app.component.css']
})
export class AppComponent {
   title = 'angular7-app';
}
```