# Useful Links

Real life educational Angular 7 Tutorials in Hindi

<https://www.truecodex.com/projects/angular/view/project-overview-and-setup>

<https://www.youtube.com/channel/UCrG9BMLvPsSJ07iHnUJq0bg/playlists>

<https://jsonplaceholder.typicode.com/>

1. Create a new application

ng new hand-world

2. Copy node-modules into root folder

3. Start an application

ng serve

in browser run localhost:4200

or

ng serve --host 0.0.0.0 --port 8589

4. Create a new command window and navigate

cd \

C:\AV\Angular4\hand-world

5. run Visual Studio Code editor in command window

code .

Shift+Ctrl+P

6. Use Ctrl+~ to open terminal to create a new component

ng g c Customers

g - means generate

c - component

Customers is a name of a new component

7.

import { CourseComponent } from './course/course.component';

8. cd into a new application folder and

cd bootstrap to the application: npm install bootstrap@3 –save

# Learning Angular

Creating a new application: ng new mosh-app

Adding bootstrap to the application: npm install bootstrap –save

It adds bootstrap import in packages.json in dependencies section. Then add bootstrap import in styles.css: @import "~bootstrap/dist/css/bootstrap.css";

Creating a component

1. Create a new file courses.component.ts in app folder

|  |
| --- |
| import { Component } from "@angular/core";  @Component({  selector: "courses",  template: `ng g c  <h2>Courses</h2>  `  })  export class CoursesComponent {} |

1. Register a new component in app.module.ts

|  |
| --- |
| import { BrowserModule } from "@angular/platform-browser";  import { NgModule } from "@angular/core";  import { AppComponent } from "./app.component";  import { CoursesComponent } from "./courses.component";  @NgModule({  declarations: [AppComponent, CoursesComponent],  imports: [BrowserModule],  providers: [],  bootstrap: [AppComponent]  })  export class AppModule {} |

1. Cd

## Creating new component using CLI

ng g c favorite

Property binding <img [src]="actrasUrl" />

Event binding <button (click)="onStarClick()">Revert</button>

### Defining Input properties

<favorite [is-emptystar]="post.isEmptyStart" (change)="onEmptyStarChange()"></favorite>

|  |
| --- |
| import { Component, OnInit, Input } from "@angular/core";  @Component({  selector: "favorite",  templateUrl: "./favorite.component.html",  styleUrls: ["./favorite.component.css"],  inputs: ['isEmptyStart']  })  export class FavoriteComponent implements OnInit {  @Input('is-emptystar') isEmptyStart: boolean = false; actrasUrl = "https://24smi.org/...";  constructor() {}  ngOnInit() {}  onStarClick() {  console.log("Star clicked");  this.isEmptyStart = !this.isEmptyStart)  }  } |

### Output properties

# HttpClient

<https://www.youtube.com/watch?v=m_jBxPn-rcU>

1. Add HttpClient into imports in app.module.ts

|  |
| --- |
| 1. import { BrowserModule } from '@angular/platform-browser'; 2. import { NgModule } from '@angular/core'; 3. import { AppComponent } from './app.component'; 4. import { HttpClientModule } from "@angular/common/http"; 5. import { PostsComponent } from './posts/posts.component'; 6. import { DataService } from './services/data.service'; 7. @NgModule({ 8. declarations: [ 9. AppComponent, PostsComponent 10. ], 11. imports: [ 12. BrowserModule, HttpClientModule 13. ], 14. providers: [DataService], 15. bootstrap: [AppComponent] 16. }) 17. export class AppModule { } |

1. Create a new service: **ng g s data** and add it into providers section inapp.module.ts

Exmple

|  |
| --- |
| import { Observable } from 'rxjs';  import { Injectable } from '@angular/core';  import { HttpClient } from '@angular/common/http';  import { T2Code } from "../models/t2code.model";  @Injectable({  providedIn: 'root'  })  export class DataService {  protected readonly URL: string = "http://localhost:8383/api/t2codes";  constructor(protected httpClient: HttpClient) { }  getData(): Observable<Array<T2Code>> {  return this.httpClient.get<Array<T2Code>>(this.URL);  }  } |

1. Create a Component: **ng g c posts**

# Creating a new node.js application (Bootstrap 4)

mkdir bs4 && cd bs4

npm init -y creating a new application without questions

npm i gulp browser-sync gulp-sass --save-dev

i – for install

gulp – javascrip task runner

gulp-sass – css compiler

Recommended to take a course on SASS.

These packages for production

npm i bootstrap jquery popper.js –save

code .

# Creating Angular Application with SASS

1. ng new data-events –style=scss
2. Notice in the file angular.json

|  |
| --- |
| "@schematics/angular:component": {  "styleext": "scss"  } |
| "styles": [  "src/styles.scss"  ], |

1. app.component.scss instead of app.component.css

The next step is to create a folder src/**scss** and move styles.scss file there.

In angular.json file change reference o the new location of the styles.scss file.

c. ====================

1. create \_variables.scss file in src/scss folder. Notice \_ "underscore" in front of the file.

2. Move $font-color: green; and $background-color: yellow; from styles.scss file into \_variables.scss file

3. in styles.scss add import: @import '\_variables';

d. =====================

1. app.component.scss is locally scoped. In order to use global \_variables.scss include import into app.component.scss file:

@import '~scss/variables'*; - this doesn’t work*

@import 'src/scss/\_variables';

'~' - means it is in src folder

2. Add a new color in the global \_variable.scss file: $other-color: #470eb1;

3. Add a new style for h2 tag in app.component.scss file using this color.

E. Installing Bootstrap

=========================

1. npm install bootstrap

check package.json file for entry with bootstrap. It should be in dependencies section

2. in styles.scss add import:

@import '~bootstrap/scss/bootstrap.scss';

You should see changes in the font on index.html

3. look at bootstrap/scss/\_variable.scss file to see what variables can be customized

F. ======================

1. install jquery and popper.js

|  |
| --- |
| npm install jquery popper.js –save |

2. Add scripts in angular.json file

|  |
| --- |
| "scripts": [  "./node\_modules/jquery/dist/jquery.slim.min.js",  "./node\_modules/popper.js/dist/umd/popper.min.js",  "./node\_modules/bootstrap/dist/js/bootstrap.min.js"  ], |

G. Adding Bootstrap & Bootwatch

================================

npm install --save bootstrap

npm install --save bootswatch

src/styles.scss

@import "~bootswatch/dist/yeti/\_variables.scss";

@import "~bootstrap/scss/bootstrap.scss";

@import "~bootswatch/dist/yeti/\_bootswatch.scss";

## Application styles.scss

|  |
| --- |
| @import '~bootstrap/scss/bootstrap.scss';  body {  padding: 10px;  margin: 5px;  } |

## Installing Font Awesome

|  |
| --- |
| // this way doesn’t work  npm install --save-dev @fortawesome/fontawesome-free  It is working way  **$ npm i --save @fortawesome/fontawesome-free-webfonts** |

Add to scss/styles.scss or to installed bootsrap.scss the following:

Check the correctness of the fortawesome path – it changes from installation to installation.

node\_modules\@fortawesome\fontawesome-free\webfonts

|  |
| --- |
| $fa-font-path: '~@fortawesome/fontawesome-free/webfonts';  @import '~@fortawesome/fontawesome-free-webfonts/scss/fontawesome.scss';  @import '~@fortawesome/fontawesome-free-webfonts/scss/fa-solid.scss';  @import '~@fortawesome/fontawesome-free-webfonts/scss/fa-regular.scss';  @import "~@fortawesome/fontawesome-free-webfonts/scss/fa-brands.scss";  @import "~@fortawesome/fontawesome-free-webfonts/scss/\_variables.scss";  @import "~@fortawesome/fontawesome-free-webfonts/scss/\_mixins.scss";  @import "~@fortawesome/fontawesome-free-webfonts/scss/\_core.scss";  @import "~@fortawesome/fontawesome-free-webfonts/scss/\_icons.scss"; |

When added to the bootstrap.scss the file extensions maybe removed

|  |
| --- |
| $fa-font-path: '~@fortawesome/fontawesome-free/webfonts';  @import '~@fortawesome/fontawesome-free/scss/fontawesome';  @import '~@fortawesome/fontawesome-free/scss/fa-solid';  @import '~@fortawesome/fontawesome-free/scss/fa-regular';  @import "~@fortawesome/fontawesome-free/scss/fa-brands";  @import "~@fortawesome/fontawesome-free/scss/\_variables";  @import "~@fortawesome/fontawesome-free/scss/\_mixins";  @import "~@fortawesome/fontawesome-free/scss/\_core";  @import "~@fortawesome/fontawesome-free/scss/\_icons"; |

List of free icons <https://fontawesome.com/icons?d=gallery&m=free>

## Using with Angular

<https://fontawesome.com/how-to-use/on-the-web/using-with/angular>

|  |
| --- |
| npm install --save-dev angular-fontawesome |

## How to use specific icons

<https://fontawesome.com/how-to-use/on-the-web/referencing-icons/basic-use>

<https://discourse.roots.io/t/how-to-use-font-awesome-5-in-your-sage-theme/11737>

## How icons named

<https://fontawesome.com/how-to-use/with-the-api/setup/importing-icons#icon-names>

## Icon names

JavaScript naming doesn’t use hyphens; it uses [lower camel case](https://en.wikipedia.org/wiki/Camel_case). So here are some basic examples of how imports and requires work.

| **Icon name** | **Version 4 name** | **Lower camel case** | **Notes on usage** |
| --- | --- | --- | --- |
| [address-book](https://fontawesome.com/icons/address-book) | fa-address-book | faAddressBook | The “B” is captialized after the hyphen |
| [facebook-f](https://fontawesome.com/icons/facebook-f) | fa-facebook | faFacebookF | It’s in the Brands style and this was renamed to facebook-f |
| [circle](https://fontawesome.com/icons/circle) | fa-circle-o | faCircle | Outline versions of icons are now in the Regular and Light styles |
| [freebsd](https://fontawesome.com/icons/freebsd) | fa-freebsd | faFreebsd | **OCD-warning**: we know it’s FreeBSD but consistency/guessability is the goal here |

Importing these specific icons:

import { faAddressBook } from '@fortawesome/free-solid-svg-icons'

import { faFacebookF } from '@fortawesome/free-brands-svg-icons'

import { faCircle } from '@fortawesome/free-regular-svg-icons'

import { faFreebsd } from '@fortawesome/free-brands-svg-icons'

Import all icons by using the prefix:

import { library } from '@fortawesome/fontawesome-svg-core'

import { fas } from '@fortawesome/free-solid-svg-icons'

import { far } from '@fortawesome/free-regular-svg-icons'

import { fab } from '@fortawesome/free-brands-svg-icons'

// Add all icons to the library so you can use it in your page

library.add(fas, far, fab)

To import the same icon from different styles:

import { library } from '@fortawesome/fontawesome-svg-core'

import { faCircle as fasFaCircle } from '@fortawesome/free-solid-svg-icons' // ES Module "as" syntax

import { faCircle as farFaCircle } from '@fortawesome/free-regular-svg-icons'

library.add(fasFaCircle, farFaCircle)

## Sizing Scale Details

| **Class** | **Size** | | **Other Notes** | |
| --- | --- | --- | --- | --- |
| fa-xs | .75em | |  | |
| fa-sm | .875em | |  | |
| fa-lg | | 1.33em | | Also applies vertical-align: -25% | |
| fa-2x | | 2em | |  | |
| fa-3x | | 3em | |  | |
| fa-4x | | 4em | |  | |
| fa-5x | | 5em | |  | |
| fa-6x | | 6em | |  | |
| fa-7x | | 7em | |  | |
| fa-8x | | 8em | |  | |
| fa-9x | | 9em | |  | |
| fa-10x | | 10em | |  | |

<head>

<link href="/your-path-to-fontawesome/css/all.css" rel="stylesheet"> <!--load all styles -->

</head>

<body>

<i class="fas fa-user"></i> <!-- uses solid style -->

<i class="far fa-user"></i> <!-- uses regular style -->

<i class="fal fa-user"></i> <!-- uses light style -->

<!--brand icon-->

<i class="fab fa-github-square"></i> <!-- uses brands style -->

</body>

## Referencing Icons

We recommend referencing icons in your HTML with a dedicated element you’ll use only for icons. We find the <i> tag perfect for the job. That element will contain: 1) Font Awesome specific style prefix (e.g. fas), and 2) the icon’s name (prefixed with fa-) you want to display.

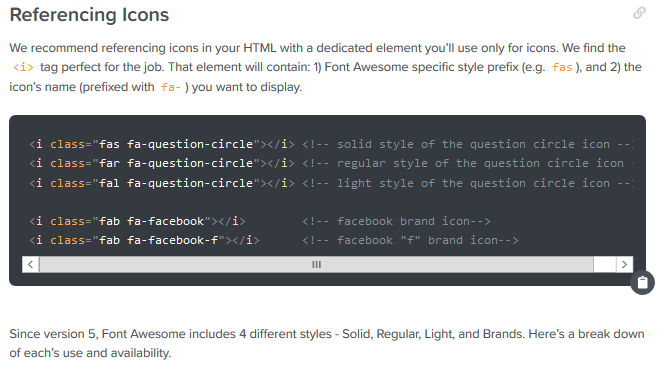
<i class="fas fa-question-circle"></i> <!-- solid style of the question circle icon -->

<i class="far fa-question-circle"></i> <!-- regular style of the question circle icon -->

<i class="fal fa-question-circle"></i> <!-- light style of the question circle icon -->

<i class="fab fa-facebook"></i> <!-- facebook brand icon-->

<i class="fab fa-facebook-f"></i> <!-- facebook "f" brand icon-->



# Creating a new application using npm fontawesome-angular

ng new bs-fa-angular --style=scss

cd bs-fa-angular

npm install bootstrap jquery popper.js –save

Modify angular.json

"styles": ["src/scss/styles.scss"],

"scripts": [

"./node\_modules/jquery/dist/jquery.slim.min.js",

"./node\_modules/popper.js/dist/umd/popper.min.js",

"./node\_modules/bootstrap/dist/js/bootstrap.min.js"

]

Add reference to bootstrap in src/scss/styles.scss file

@import '~bootstrap/scss/bootstrap.scss';

Creating Angular 6 + Bootstrap + sass + jQuery + Popper.js and Angular-fortawesome fonts

<https://fontawesome.com/how-to-use/on-the-web/using-with/angular>

<https://www.npmjs.com/package/@fortawesome/angular-fontawesome>

<https://github.com/FortAwesome/angular-fontawesome#examples>

|  |
| --- |
| Some other considerations:   1. angular-fontawesome integrates tightly with the Angular framework, so if you think you may want to have some data bound icons (eg: email icon with a counter of unread messages, or some icon with a rotation angle bound to some data value in your model), then you may find it nice to use angular-fontawesome where the icon is not only rendered as an SVG, it's rendered as an Angular component. 2. The SVG method of Font Awesome 5 is capable of some features that are not available with the Web Fonts and CSS method. For example, [Power Transforms](https://fontawesome.com/how-to-use/on-the-web/styling/power-transforms), [Layers, and Counters](https://fontawesome.com/how-to-use/on-the-web/styling/layering). Since angular-fontawesome builds on the SVG with JavaScript implementation, it supports all of those extra features too. 3. Tree-shaking / subsetting: There are many, many icons that you won't be using. If you care about optimizing the size of your downloaded assets in your production builds, then you might want to make sure you're using tree-shaking, which you'll get [mostly automagically](https://fontawesome.com/how-to-use/with-the-api/other/tree-shaking) if you use angular-cli (Caveat: see that link about configuring your build to work around a build performance regression that makes Webpack 4 production builds slow until the fix is released.) |

1. install wget with dependencies
2. install yarn
3. Add required modules into angular using yarn as described in this document <https://www.npmjs.com/package/@fortawesome/angular-fontawesome> v.4

<https://github.com/FortAwesome/angular-fontawesome> v.5 for angular 5

|  |
| --- |
| $ yarn add @fortawesome/fontawesome-svg-core  $ yarn add @fortawesome/free-solid-svg-icons  $ yarn add @fortawesome/angular-fontawesome |

1. Add imports
2. Add icons to the application

<https://github.com/FortAwesome/angular-fontawesome#examples>

<https://fontawesome.com/how-to-use/on-the-web/advanced/svg-javascript-core>

List of free icons <https://fontawesome.com/icons?d=gallery&m=free>

How to change size of icons

<https://github.com/FortAwesome/angular-fontawesome>

# Configuring your project to use yarn instead of npm

Source: <https://theinfogrid.com/tech/developers/angular/setup-angular-touse-yarn-package-manager/>

ng new directives2 --style=scss

cd template-forms

ng config -g cli.packageManager yarn

ng config -g cli.packageManager npm

yarn add bootstrap

yarn add jquery

yarn add popper.js

yarn add @fortawesome/fontawesome-svg-core

yarn add @fortawesome/free-solid-svg-icons

yarn add @fortawesome/angular-fontawesome

yarn add @fortawesome/free-brands-svg-icons

yarn add @fortawesome/free-regular-svg-icons

yarn init

npm install bootstrap jquery popper.js --save

npm install @fortawesome/fontawesome-svg-core –save

npm install @fortawesome/free-solid-svg-icons -–save

npm install @fortawesome/angular-fontawesome -–save

npm install @fortawesome/free-brands-svg-icons -–save

npm install @fortawesome/free-regular-svg-icons -–save

October 14, 2018

Template Forms with Mosh

ng new tempalte-forms --style=scss

ng config -g cli.packageManager npm

cd template-forms

npm install bootstrap jquery popper.js –save

npm install @fortawesome/angular-fontawesome -–save

npm install @fortawesome/fontawesome-svg-core -–save

npm install @fortawesome/free-solid-svg-icons -–save

npm install @fortawesome/free-brands-svg-icons -–save

npm install @fortawesome/free-regular-svg-icons -–save

code .

Create folder in src/scss

Move styles.scss into src/scss

Change angular.json file to accommodate this change

Add the following into scripts section in angular.json file

|  |
| --- |
| "./node\_modules/jquery/dist/jquery.slim.min.js",  "./node\_modules/popper.js/dist/umd/popper.min.js",  "./node\_modules/bootstrap/dist/js/bootstrap.min.js" |

# Template Forms with Bootstrap 4, Fontawesome

October 16, 2018

1. ng new ng-sass --style=scss
2. cd ex-template-forms

ng config -g cli.packageManager npm

1. npm install bootstrap jquery popper.js –save
2. npm install @fortawesome/angular-fontawesome –save
3. npm install @fortawesome/fontawesome-svg-core -–save
4. npm install @fortawesome/free-solid-svg-icons -–save
5. npm install @fortawesome/free-brands-svg-icons -–save
6. npm install @fortawesome/free-regular-svg-icons -–save

code .

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

Then, add into styles.scss the following entries pointing to bootstrap

|  |
| --- |
| @import '~bootstrap/scss/bootstrap.scss';  @import '\_variables';  body {  background-color: $background-color;  } |

Create a \_variables.scss file in src/scss folder and add examples of custom colors

|  |
| --- |
| $font-color: green;  $background-color:rgb(228, 240, 235);  $other-color: #f76464; |

Add imports into app.moduls.ts file

|  |
| --- |
| import { FormsModule } from "@angular/forms";  import { FontAwesomeModule } from "@fortawesome/angular-fontawesome"; |

Creating a new form component i.e. reactive-form

ng g c reactive-form

Add imports into your form

|  |
| --- |
| @import "~bootstrap/scss/bootstrap.scss";  @import "src/scss/\_variables";  h2 {  color: $other-color;  }  .main-form {  width: 500px;  margin: auto;  }  .form-control.ng-touched.ng-invalid {  border: 2px solid red;  } |

Add references to icons and register them as library for the form

|  |
| --- |
| import { faCoffee, faStarOfDavid, faCheck } from '@fortawesome/free-solid-svg-icons';  import { library } from '@fortawesome/fontawesome-svg-core';  import { fas } from '@fortawesome/free-solid-svg-icons';  import { far } from '@fortawesome/free-regular-svg-icons';  import { fab } from '@fortawesome/free-brands-svg-icons';  library.add(fas, far, fab);  library.add(faCoffee);  library.add(faStarOfDavid);  library.add(faCheck); |

Also in the component class add variables for your icons i.e.

|  |
| --- |
| faStarOfDavid = faStarOfDavid;  faCoffee = faCoffee;  faCheck = faCheck; |

List of free available icons can be found here <https://fontawesome.com/icons?d=gallery&m=free>

I component html add example of icons use i.e.

|  |
| --- |
| <button class="btn btn-primary" [disabled]="!f.valid">  <div style="text-align:left">  <fa-icon [icon]="['fas', 'check']" size="3x"></fa-icon>  </div>Coffee?  </button> |

# ANGULAR 7

To update 6 to 7 run:

ng update @angular/cli @angular/core

## Angular Update Guide | 6.1 -> 7.0 for Basic Apps

### Before Updating



Remove deprecated RxJS 6 features using [rxjs-tslint auto update rules](https://github.com/ReactiveX/rxjs-tslint)  
  
For most applications this will mean running the following two commands:  
  
npm install -g rxjs-tslint  
rxjs-5-to-6-migrate -p src/tsconfig.app.json

### During the Update



Angular now uses TypeScript 3.1, [read more](https://www.typescriptlang.org/docs/handbook/release-notes/typescript-3-1.html) about any potential breaking changes



Angular has now added support for [Node 10](https://nodejs.org/en/blog/release/v10.0.0/)



Update to v7 of the core framework and CLI by running ng update @angular/cli @angular/core in your terminal

### After the Update

There aren't currently any changes needed after moving between these versions.

# HttpClientModule

# 

<https://update.angular.io>

<https://www.academind.com/learn/javascript/rxjs-6-what-changed/>