Exercise (Instructions): Setting up Git

Objectives and Outcomes

In this exercise you will learn to install Git on your computer. Git is required for using all the remaining Node.js and Node based tools that we encounter in the rest of the course. At the end of this exercise, you would be able to:

* Install Git on your computer
* Ensure that Git can be used from the command-line or command-prompt on your computer
* Set up some of the basic global configuration for Git

Downloading and Installing Git

* To install Git on your computer, go to <https://git-scm.com/downloads> to download the Git installer for your specific computing platform.
* Then, follow the installation steps as you install Git using the installer.
* You can find more details about installing Git at <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>. This document lists several ways of installing Git on various platforms.
* Installing some of the GUI tools like GitHub Desktop will also install Git on your computer.
* On a Mac, setting up XCode command-line tools also will set up Git on your computer.
* You can choose any of the methods that is most convenient for you.

Some Global Configuration for Git

* Open a cmd window or terminal on your computer.
* Check to make sure that Git is installed and available on the command line, by typing the following at the command prompt:

git –version

* To configure your user name to be used by Git, type the following at the prompt:

git config --global user.name "Your Name"

* To configure your email to be used by Git, type the following at the prompt:

git config --global user.email <your email address>

* You can check your default Git global configuration, you can type the following at the prompt:

git config –list

# Exercise (Instructions): Basic Git Commands

### Objectives and Outcomes

In this exercise you will get familiar with some basic Git commands. At the end of this exercise you will be able to:

* Set up a folder as a Git repository
* Perform basic Git operations on your Git repository

## **Basic Git Commands**

* At a convenient location on your computer, create a folder named **git-test**.
* Open this git-test folder in your favorite editor.
* Add a file named index.html to this folder, and add the following HTML code to this file:

<!DOCTYPE html>

<html>

<head></head>

<body>

<h1>This is a Header</h1>

</body>

</html>

Initializing the folder as a Git repository

* Go to the git-test folder in your cmd window/terminal and type the following at the prompt to initialize the folder as a Git repository:

git init

Checking your Git repository status

* Type the following at the prompt to check your Git repository's status:

git status

Adding files to the staging area

* To add files to the staging area of your Git repository, type:

git add .

Commiting to the Git repository

* To commit the current staging area to your Git repository, type:

git commit -m "first commit"

Checking the log of Git commits

* To check the log of the commits to your Git repository, type

git log –oneline

* Now, modify the*index.html*file as follows:

<!DOCTYPE html>

<html>

<head></head>

<body>

<h1>This is a Header</h1>

<p>This is a paragraph</p>

</body>

</html>

* Add a sub-folder named **templates** to your **git-test** folder, and then add a file named *temp.html* to the templates folder. Then set the contents of this file to be the same as the *index.html* file above.
* Then check the status and add all the files to the staging area.
* Then do the second commit to your repository
* Now, modify the*index.html*file as follows:

<!DOCTYPE html>

<html>

<head></head>

<body>

<h1>This is a Header</h1>

<p>This is a paragraph</p>

<p>This is a second paragraph</p>

</body>

</html>

* Now add the modified index.html file to the staging area and then do a third commit.

Checking out a file from an earlier commit

* To check out the index.html from the second commit, find the number of the second commit using the git log, and then type the following at the prompt:

git checkout <second commit's number> index.html

Resetting the Git repository

* To discard the effect of the previous operation and restore index.html to its state at the end of the third commit, type:

git reset HEAD index.html

* Then type the following at the prompt:

git checkout -- index.html

* You can also use *git reset* to reset the staging area to the last commit without disturbing the working directory.

Exercise (Instructions): Online Git Repositories

Objectives and Outcomes

In this exercise you will learn about how to set up and use an online Git repository and synchronize your local Git repository with your online repository. At the end of this exercise, you will be able to:

* Set up the online repository as a remote repository for your local Git repository
* Push your commits to the online repository
* Clone an online Git repository to your computer

Setting up an Online Git repository

* Sign up for an account either at Bitbucket ([https://bitbucket.org](https://bitbucket.org/)) or GitHub ([https://github.com](https://github.com/)). Note that private repositories on GitHub requires a paid account, and is not available for free accounts.
* Then set up an online Git repository named **git-test**. Note the URL of your online Git repository.

Set the local Git repository to set its remote origin

* At the prompt, type the following to set up your local repository to link to your online Git repository:

git remote add origin <repository URL>

Pushing your commits to the online repository

* At the prompt, type the following to push the commits to the online repository:

git push -u origin master

Cloning an online repository

* To clone an online repository to your computer, type the following at the prompt:

git clone <repository URL>

Exercise (Instructions): Setting up Node.js and NPM

**Note: Make sure you have installed Git on your machine before you install Node.js. Please complete the previous Git installation exercise before proceeding with this exercise.**

Objectives and Outcomes

In this exercise, you will learn to set up the Node.js environment, a popular Javascript based server framework, and node package manager (NPM) on your machine. To learn more about NodeJS, you can visit [https://nodejs.org](https://nodejs.org/). For this course, you just need to install Node.js on your machine and make use of it for running some front-end tools. You will learn more about the server-side support using Node.js in a subsequent course. At the end of this exercise, you will be able to:

* Complete the set up of Node.js and NPM on your machine
* Verify that the installation was successful and your machine is ready for using Node.js and NPM.

Installing Node

* To install Node on your machine, go to [https://nodejs.org](https://nodejs.org/) and click on the Download button. Depending on your computer's platform (Windows, MacOS or Linux), the appropriate installation package is downloaded.
* As an example, on a Mac, you will see the following web page. Click on the Download button. Follow along the instructions to install Node on your machine. (Note: Now Node gives you the option of installing a mature and dependable LTS version and a more newer stable version. You should to install the LTS version. I will use this version in the course.)

**Note: On Windows machines, you may need to configure your PATH environmental variable in case you forgot to turn on the add to PATH during the installation steps.**

Verifying the Node Installation

* Open a terminal window on your machine. If you are using a Windows machine, open a cmd window or PowerShell window with **admin** privileges.
* To ensure that your NodeJS setup is working correctly, type the following at the command prompt to check for the version of **Node** and **NPM**

node -v

npm –v

Exercise (Instructions): Basics of Node.js and NPM

Objectives and Outcomes

In this exercise you will learn the basics of Node and NPM. At the end of this exercise, you will be able to:

* Set up package.json file in the project folder for configuring your Node and NPM for this project
* Install a NPM module and make use of it within your project

Initializing package.json

* At the command prompt in your **git-test** folder, type

npm init

* Follow along the prompts and answer the questions as follows: accept the default values for most of the entries, except set the entry point to index.html
* This should create a *package.json* file in your **git-test** folder.

Installing an NPM Module

* Install an NPM module, lite-server, that allows you to run a Node.js based development web server and serve up your project files. To do this, type the following at the prompt:

npm install lite-server --save-dev

* You can check out more documentation on lite-server [here](https://github.com/johnpapa/lite-server).
* Next, open package.json in your editor and modify it as shown below. Note the addition of two lines, line 7 and line 9.

{

"name": "git-test",

"version": "1.0.0",

"description": "This is the Git and Node basic learning project",

"main": "index.html",

"scripts": {

"start": "npm run lite",

"test": "echo \"Error: no test specified\" && exit 1",

"lite": "lite-server"

},

"repository": {

"type": "git",

"url": "git+https://jogesh\_k\_muppala@bitbucket.org/jogesh\_k\_muppala/git-test.git"

},

"author": "",

"license": "ISC",

"homepage": "https://bitbucket.org/jogesh\_k\_muppala/git-test#readme",

"devDependencies": {

"lite-server": "^2.2.2"

}

}

* Next, start the development server by typing the following at the prompt:

npm start

* This should open your *index.html* page in your default browser.
* If you now open the *index.html* page in an editor and make changes and save, the browser should immediately refresh to reflect the changes.

Setting up .gitignore

* Next, create a file in your project directory named *.gitignore* (**Note**: the name starts with a period)Then, add the following to the .gitignore file

node\_modules

* The do a git commit and push the changes to the online repository. You will note that the node\_modules folder will not be added to the commit, and will not be uploaded to the repository.

Exercise (Instructions): Getting Started with Angular

Objectives and Outcomes

In this first Angular exercise, you will first install *angular-cli*, the command line tool for scaffolding Angular applications. You will then use the tool to scaffold out a basic Angular application. We will thereafter develop this application into a full-fledged Angular application in the process of doing the exercises in this course. At the end of this exercise you will be able to:

* Install *angular-cli*
* Scaffold out a basic Angular application

Installing *Angular-CLI*

From the Angular-CLI documentation we learn that the Angular CLI makes it easy to create an application that already works, right out of the box. It already follows the best practices suggested by the Angular community!

* To install *angular-cli* globally, type the following at the prompt:

npm install -g @angular/cli

Use *sudo* on a Mac and Linux

* This will make the command line tool for creating Angular applications. To learn more about the various commands that this CLI provides, type at the prompt:

ng help

Generating and Serving an Angular Project using Angular-CLI

* At a convenient location on your computer, create a folder named *Angular* and move into that folder.
* Then type the following at the prompt to create a new Angular application named *conFusion*:

ng new conFusion -dir=<The path of your Angular folder>/conFusion --style=scss

* This should create a new folder named *conFusion* within your *Angular* folder and create the Angular application in that folder.
* Move to the conFusion folder and type the following at the prompt:

npm install

ng serve –open

* This will compile the project and then open a tab in your default browser at the address <http://localhost:4200>.
* You can initialize your project to be a Git repository by typing the following commands at the prompt:

git init

git add .

git commit -m "Initial Setup"

**Note**: Some of you may find that Angular CLI automatically does the first commit on your computer and initializes the Git repository. Please do a "git status" in the project folder just to check if an automatic commit has been done. This doesn't happen on my computer. Hence the above instructions.

* Thereafter you can set up an online Git repository and synchronize your project to the online repository. Make sure that the online Git repository is a ***private*** repository.

Exercise (Instructions): Configuring your Angular Application

Objectives and Outcomes

In this exercise we will set up our project to use Angular Material and Angular Flex Layout. We will then introduce our first Angular Material component into our application. At the end of this exercise you will be able to:

* Configure your Angular project to use Angular Material and Flex Layout.
* Start using Material components in your application.

Configure your Angular Project to use Angular Material

Note: This course is designed with Angular Material Beta.3. Before you proceed forward, you may wish to read the detailed information posted in <https://www.coursera.org/learn/angular/discussions/all/threads/4yxVk7DXEee0mQrUfDuicA> where I have clearly explained about dealing with the newer Beta versions of Angular Material (up to Beta.12). I would strongly suggest that to proceed ahead with the course with minimal disruption, please install the Beta.8 version of Angular Material. With this installation, the course instructions will still work as given.

* To configure your project to use Angular material, type the following at the prompt to install Angular Material, Angular Animations and HammerJS:

npm install @angular/material@2.0.0-beta.8 --save

npm install @angular/cdk@2.0.0-beta.8 --save

npm install --save @angular/animations

npm install --save hammerjs

Configure to use Material Design Icons

* Next, include the following into the <head> of *index.html* to make use of Material Design icons:

<link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">

Configure your Angular Project to use Flex Layout

* Next, install Angular Flex Layout as follows: npm install --save @angular/flex-layout@latest

Updating AppModule

* Then, you need to import the Angular Animations Module, Angular Material Module, Flex Layout Module and hammerjs into your root module (src/app/app.module.ts) as follows:

. . .

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

import { MaterialModule } from '@angular/material';

import { FlexLayoutModule } from '@angular/flex-layout';

. . .

import 'hammerjs';

@NgModule({

. . .

imports: [

. . .,

BrowserAnimationsModule,

MaterialModule,

FlexLayoutModule

],

. . .

})

. . .

Adding a Material Toolbar

* Open app.component.html and replace its contents with the following code:

<md-toolbar color="primary"> <span>Ristorante Con Fusion</span> </md-toolbar>

Adding Styles

* Add the following styles to styles.scss file:

@import '~@angular/material/prebuilt-themes/deeppurple-amber.css';

// some basic resets

body {

padding: 0;

margin: 0;

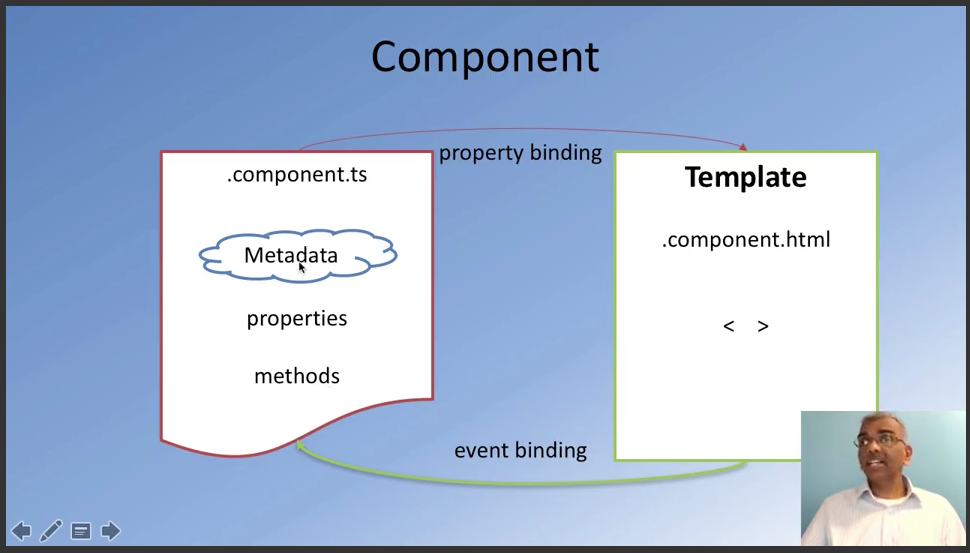
font-family: Roboto, sans-serif;

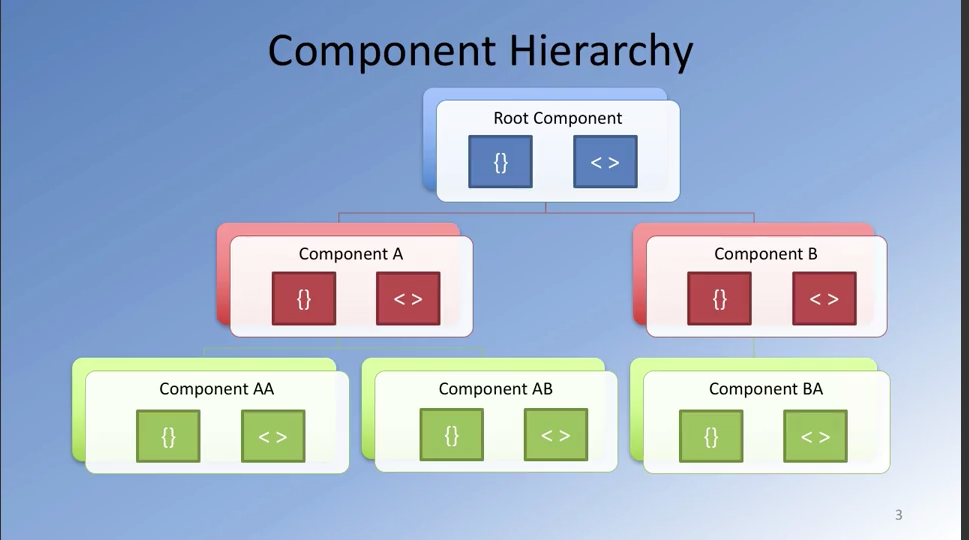
}

* This will add a built-in Material theme to our application.
* Do a Git commit with the message "Configuring Angular"

Conclusions

In this exercise we learnt to use Angular Material and Flex Layout NgModules in our Angular application.





# Exercise (Instructions): Angular Components Part 1

### Exercise Resources

### Download image

Objectives and Outcomes

In this exercise you will add the first component to your Angular application and update its template. At the end of this exercise you will be able to:

* Add components to your Angular application
* Update the templates of your component.

Adding a Menu Component

* First, download the images.zip file provided above and then unzip the file. Move the resulting *images* folder containing some PNG files to the Angular project's *src/assets* folder. These image files will be useful for our exercises.
* Next, use the CLI's *ng generate* command to generate a new component named menu as follows:

### ng generate component menu

* This will create the necessary files for the menu component in a folder named *menu*, and also import this component into *app.module.ts*.
* Next, open app.component.html file and add the following after the toolbar:

Creating the Menu

* Next, create a folder named *shared* under the *src/app* folder. To this folder, add a file named dish.ts with the following code:

### export class Dish {

### name: string;

### image: string;

### category: string;

### label: string;

### price: string;

### description: string;

### }

* Update menu.component.ts as follows to add in the data for four menu items:

### . . .

### import { Dish } from '../shared/dish';

### . . .

### export class MenuComponent implements OnInit {

### dishes: Dish[] = [

### {

### name:'Uthappizza',

### image: '/assets/images/uthappizza.png',

### category: 'mains',

### label:'Hot',

### price:'4.99',

### description:'A unique combination of Indian Uthappam (pancake) and Italian pizza, topped with Cerignola olives, ripe vine cherry tomatoes, Vidalia onion, Guntur chillies and Buffalo Paneer.' },

### {

### name:'Zucchipakoda',

### image: '/assets/images/zucchipakoda.png',

### category: 'appetizer',

### label:'',

### price:'1.99',

### description:'Deep fried Zucchini coated with mildly spiced Chickpea flour batter accompanied with a sweet-tangy tamarind sauce' },

### {

### name:'Vadonut',

### image: '/assets/images/vadonut.png',

### category: 'appetizer',

### label:'New',

### price:'1.99',

### description:'A quintessential ConFusion experience, is it a vada or is it a donut?' },

### {

### name:'ElaiCheese Cake',

### image: '/assets/images/elaicheesecake.png',

### category: 'dessert',

### label:'',

### price:'2.99',

### description:'A delectable, semi-sweet New York Style Cheese Cake, with Graham cracker crust and spiced with Indian cardamoms' }

### ];

### . . .

### }

* Next, update the menu.component.html template as follows:

### <div class="container"

### fxLayout="column"

### fxLayoutGap="10px">

### <md-list fxFlex>

### <md-list-item \*ngFor="let dish of dishes">

### <img md-list-avatar src={{dish.image}} alt={{dish.name}}>

### <h1 md-line> {{dish.name}} </h1>

### <p md-line>

### <span> {{dish.description}} </span>

### </p>

### </md-list-item>

### </md-list>

### </div>

* Add the following CSS class to styles.scss file:

### .container {

### margin: 20px;

### display:flex;

### }

* Save all changes and do a Git commit with the message "Components Part 1".

Exercise (Instructions): Angular Components Part 2

Objectives and Outcomes

In this exercise we will continue modifying the component template from the previous exercise. Instead of a list, we will use a grid list Angular material component to display the menu in a different way. Also we will use the Card component to display the details of a selected dish. At the end of this exercise you will be able to:

* Make use of the Angular material grid list component to display a list of items.
* Use the material Card component to display detailed information.
* Use a built-in Angular pipe to turn a word into uppercase in the template.

Updating the Menu Template

* Open *menu.component.html* and update its content as follows:

### <div class="container"

### fxLayout="column"

### fxLayoutGap="10px">

### <div fxFlex>

### <div>

### <h3>Menu</h3>

### <hr>

### </div>

### </div>

### <div fxFlex>

### <md-grid-list cols="2" rowHeight="200px">

### <md-grid-tile \*ngFor="let dish of dishes">

### <img height="200px" src={{dish.image}} alt={{dish.name}}>

### <md-grid-tile-footer>

### <h1 md-line>{{dish.name | uppercase}}</h1>

### </md-grid-tile-footer>

### </md-grid-tile>

### </md-grid-list>

### </div>

### </div>

* Here we are using the Grid list Angular material component to display the information.
* Also, update the *menu.component.ts* file as follows to move the details of the dishes into a constant, in preparation for introducing services in a future exercise:

### const DISHES: Dish[] = [

### . . .

### 

### ];

### 

### . . .

### 

### export class MenuComponent implements OnInit {

### dishes = DISHES;

### selectedDish = DISHES[0];

### . . .

### 

### }

Add a Card Component

* Update the menu.component.html template to display the details of a selected dish using the Material Card component as follows:

### <div fxFlex \*ngIf="selectedDish">

### <md-card>

### <md-card-header>

### <md-card-title>

### <h3>{{selectedDish.name | uppercase}}</h3>

### </md-card-title>

### </md-card-header>

### <img md-card-image src={{selectedDish.image}} alt={{selectedDish.name}}>

### <md-card-content>

### <p>{{selectedDish.description}}

### </p>

### </md-card-content>

### <md-card-actions>

### <button md-button>LIKE</button>

### <button md-button>SHARE</button>

### </md-card-actions>

### </md-card>

### </div>

* Save the changes and do a Git commit with the message "Components Part 2".

Conclusions

In this exercise we used a grid list to display the information in the menu template. Also we used a card to display the details of a selected dish.

### Usefull links:

### Angular component: <https://angular.io/guide/architecture#!%23components>

### Component style: <https://angular.io/guide/component-styles>

### Angular template: <https://angular.io/guide/architecture#!%23components>

### Metadata: <https://angular.io/guide/architecture#!%23components>

### Directive: <https://angular.io/guide/architecture#!%23components>

### Structural directive: <https://angular.io/guide/structural-directives>

### Pipes: <https://angular.io/guide/pipes>

### Material list: <https://material.angular.io/components/list/overview>

### Material grid list: <https://material.angular.io/components/grid-list/overview>

### Material card; <https://material.angular.io/components/card/overview>

### Material button: <https://material.angular.io/components/button/overview>

## **Peer-graded Assignment: Angular Components**

**You submitted!**

Your work is ready to be reviewed by classmates. Next, you need to review your classmates' work. We'll email you when your grade is ready. Your grade should be ready by **January 3, 11:59 PM PST**.

In this assignment you will add a new component to the Angular application to show the details of a selected dish. You will use the Angular Material card component and the list component to prepare the template for this new component.

**Step-By-Step Assignment Instructionsless**

**Objectives and Outcomes**

In this assignment, you will continue to work with the Angular application that you have been developing in the exercises. You will add a new component named ***dishdetail*** that will display the details of a selected dish. You will then design the template for the component using Angular material components. At the end of this assignment, you should have completed the following tasks:

* Created a new dishdetail component and added it to your Angular application and included it into the template of the menu component.
* Updated the template of the dishdetail component to display the details of the selected dish using an Angular card component.
* Updated the template of the dishdetail component to display the list of comments about the dish using the Angular material list component.

**Assignment Requirements**

This assignment requires you to complete the following tasks. Detailed instructions for each task are given below. The picture of the completed web page included below indicates the location within the web page that will be updated by the three tasks.

**Task 1**

In this task you will be adding a new ***dishdetail*** component to your Angular application and include the component into the menu component's template so that the details of a specific dish are displayed there:

* Use Angular CLI to create a new component named ***dishdetail***,
* Replace the card showing the selected dish in menu component's template with the dishdetail component, and
* Update the template of the dishdetail component with the following code:

### <div class="container"

### fxLayout="row"

### fxLayout.sm="column"

### fxLayout.xs="column"

### fxLayoutAlign.gt-md="space-around center"

### fxLayoutGap="10px"

### fxLayoutGap.xs="0">

### <div fxFlex="40">

### <p>Display the details of the Dish here</p>

### </div>

### <div fxFlex="40">

### <p>Display the list of comments here</p>

### </div>

### </div>

**Task 2**

In this task you will be adding a card component to the dishdetail template to display the details of the dish given above:

* Add a new constant to the dishdetail.component.ts file named DISH as follows, and initialize it to the JavaScript object given below that contains the details of the dish and comments about the dish:

### const DISH = {

### name: 'Uthappizza',

### image: '/assets/images/uthappizza.png',

### category: 'mains',

### label: 'Hot',

### price: '4.99',

### description: 'A unique combination of Indian Uthappam (pancake) and Italian pizza, topped with Cerignola olives, ripe vine cherry tomatoes, Vidalia onion, Guntur chillies and Buffalo Paneer.',

### comments: [

### {

### rating: 5,

### comment: "Imagine all the eatables, living in conFusion!",

### author: "John Lemon",

### date: "2012-10-16T17:57:28.556094Z"

### },

### {

### rating: 4,

### comment: "Sends anyone to heaven, I wish I could get my mother-in-law to eat it!",

### author: "Paul McVites",

### date: "2014-09-05T17:57:28.556094Z"

### },

### {

### rating: 3,

### comment: "Eat it, just eat it!",

### author: "Michael Jaikishan",

### date: "2015-02-13T17:57:28.556094Z"

### },

### {

### rating: 4,

### comment: "Ultimate, Reaching for the stars!",

### author: "Ringo Starry",

### date: "2013-12-02T17:57:28.556094Z"

### },

### {

### rating: 2,

### comment: "It's your birthday, we're gonna party!",

### author: "25 Cent",

### date: "2011-12-02T17:57:28.556094Z"

### }

### ]

### };

**NOTE**: Do not use the Dish type from the dish.ts file to declare either the const DISH or the variable dish below to be of the type Dish. We need to update the Dish type which will be done in the next module.

* Now introduce a new variable in the dishdetail.component.ts file in the dishdetail class called dish and set it equal to the DISH constant above:

### dish = DISH;

* The Angular material card component should be used to display the details of the dish as shown above. Please remember to use the Angular "uppercase" pipe on the name displayed in the card title. Also apply the \*ngIf="dish" structural directive to both the <md-card> that displays the details of the dish.

**Task 3**

In this task you will use the comments that are included in the dish object above to display a list of the comments for the dish. Please use your JavaScript knowledge to recall how you would access an inner property in a JavaScript object that itself points to an array of JavaScript objects (comments). This task involves the following steps:

* Use the Angular material list to display the list of comments as shown below. Also apply the \*ngIf="dish" structural directive to both the <md-list> that displays the list of comments for the dish.
* Display the date of the comment by processing it through the Angular built-in *date* pipe.