1. Although [NgModel](https://angular.io/api/forms/NgModel) is a valid Angular directive, it isn't available by default. It belongs to the optional [FormsModule](https://angular.io/api/forms/FormsModule). You must opt-in to using that module.
2. Immediately I save the TS file corresponding JS and js map file will be created.
3. I can’t use “const” keyword inside a class i.e.

Export class ClassName{

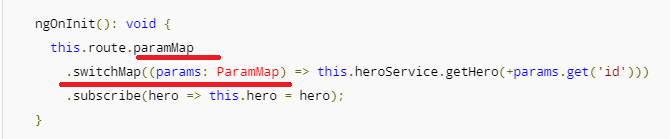
}

1. An simple example on how to compare in Angular 2is 🡺\*ngIf = ”obj1===obj2”

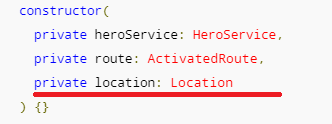
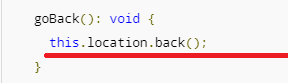
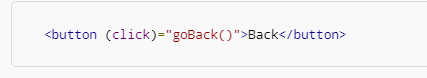
|  |  |
| --- | --- |
|  | By default, an arrow function () => {...} has the return type void unless you return something. |

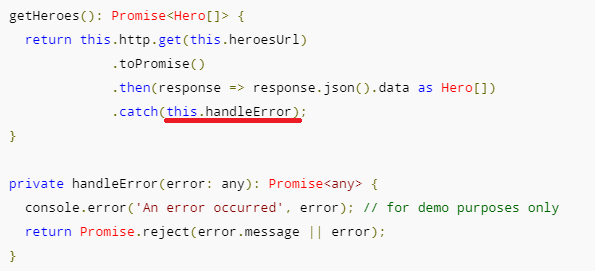
1. import { [Component](https://angular.io/api/core/Component), [Input](https://angular.io/api/core/Input) } from '@angular/core';
2. **Examples on @Input we will see this once after seeing Session notes**
3. While Using the Promise I got the Following error in the console🡺

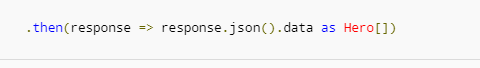
**“Cannot find a differ supporting object ‘[object Promise]’ of type ‘object’ . NgFor only supports binding to iterables such as arrays”.**

1. Multi Component Chapter has 🡺 @Input
2. Service Chapter has🡺 life cycle hooks
3. The Angular router is an external, optional Angular NgModule called [RouterModule](https://angular.io/api/router/RouterModule).
4. ([RouterOutlet](https://angular.io/api/router/RouterOutlet), [RouterLink](https://angular.io/api/router/RouterLink), [RouterLinkActive](https://angular.io/api/router/RouterLinkActive))🡺 all are directive it seems
5. Needed more information on the [ParamMap](https://angular.io/api/router/ParamMap) 🡺 this is present in the Routing chapter
6. import 'rxjs/add/operator/**switchMap'**🡺 this is present in the Routing chapter
7. example on SwitchMap and ParamMap is🡺
8. Do you need to unsubscribe?

The subscriptions are cleaned up when the component is destroyed, protecting against memory leaks, so you don't need to unsubscribe from the route paramMap Observable.

1. **Routing was not working properly for me.**
2. Now add an option, a goBack() method that navigates backward one step in the browser's history stack using the [Location](https://angular.io/api/common/Location) service.
3. import { [Location](https://angular.io/api/common/Location) } from '@angular/common';
4. 
5. 
6. 
7. The [HttpModule](https://angular.io/api/http/HttpModule) is not a core NgModule. [HttpModule](https://angular.io/api/http/HttpModule) is Angular's optional approach to web access. It exists as a separate add-on module called @angular/http
8. **Needed more explanation on InMemoryWebApiModule and InMemoryDbService**
9. Example on catch operator



1. The Angular http.get returns an RxJS Observable. Observables are a powerful way to manage asynchronous data flows
2. We can converted the Observable to a Promise using the toPromise operator. 🡺 import 'rxjs/add/operator/toPromise';
3. 
4. Each [Http](https://angular.io/api/http/Http) service method returns an Observable of HTTP [Response](https://angular.io/api/http/Response) objects.
5. toPromise operator to the Observable result of http.get()converted the Observable into a Promise and you passed that promise back to the caller.
6. Observables are to good for Asynchronous response
7. A request-cancel-new-request sequence is difficult to implement with Promises, but easy with Observables.
8. export class InMemoryDataService\_MyCUSTOMCLass implements InMemoryDbService 🡺 suppose if I change the method bame to createDB1 I get the following error in the editor

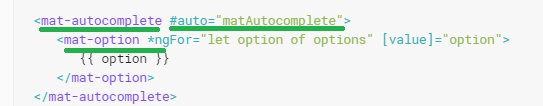
“[ts] class InMemoryDataService\_ MyCUSTOMCLass incompletely implements interface InMemoryDbService property cretaeDB is missing in type InMemoryDataService\_MyCUSTOMCLass”

1. **Angular apps are modular** and Angular has its own modularity system called NgModules.
2. Every Angular app has at least one NgModule class, [the root module](https://angular.io/guide/bootstrapping), conventionally named AppModule
3. While the root module may be the only module in a small application, most apps have many more feature modules,
4. **Angular has many decorators that attach metadata to classes so that it knows what those classes mean and how they should work**
5. The NgModule — a class decorated with @[NgModule](https://angular.io/api/core/NgModule)
6. **Data binding is also important for communication between parent and child components.**
7. **Directives** 🡺Angular templates are dynamic. When Angular renders them, it transforms the DOM according to the instructions given by **directives**.

# **Angular Material Design**

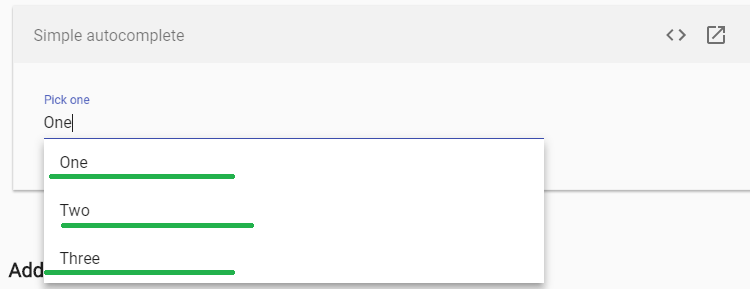
1. **Autocomplete**
2. The autocomplete is a normal input text box enhanced by a panel of suggested options
3. First lets see how to create an angular based input field



1. Now let’s see the syntax for Autocomplete
2. Now let’s see how to add this autocomplete panel to the Input text field
3. Output: 🡺

Part 1:initially🡺

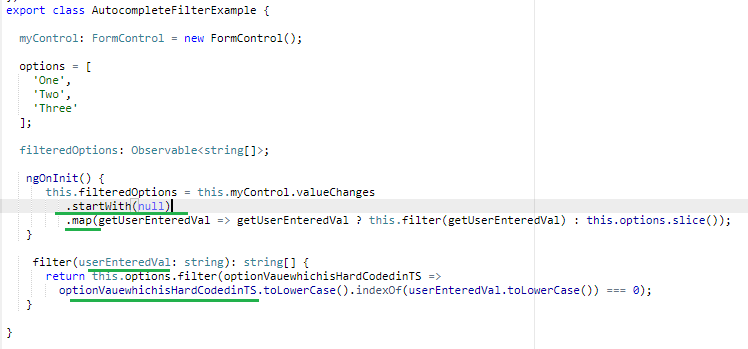
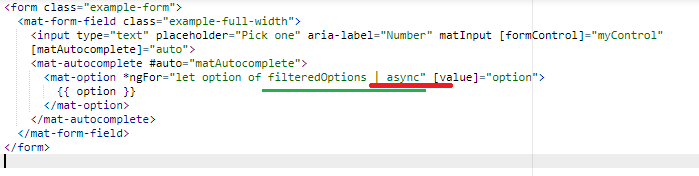
Part2:While placing the Cursor in the text box

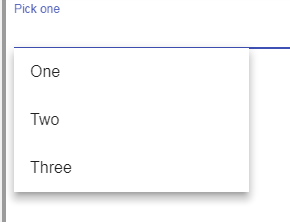


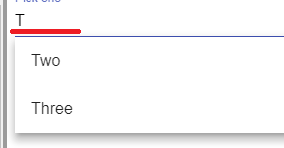
### Part2 of AutoComplete 🡺 Adding a custom filter

1. if we want our options to filter when we type, we need to add a custom filter.
2. Here we will perform a simple string test on the option value to see if it matches the input value, starting from the option's first letter
3. Needed more information on valueChanges observable ( in our example we had added it on the FormControl)
4. The resulting observable (filteredOptions) can be added to the template in place of the options property using the async pipe.
5. If you want the option's control value (what is saved in the form) to be different than the option's display value (what is displayed in the actual text field), you'll need to set the **displayWith** property on your autocomplete element.
6. For Practical explanation🡺 I will provide this later once I get the full concept

**My Understanding on Custom Filter**

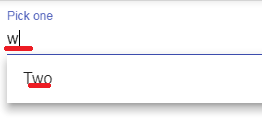
1. 
2. 
3. Explanation 🡺the flow goes as below
4. Initially filteredOptions which is an Observable of type String array is Empty
5. On Component loading/ page loading ngOnInit() method is called in which filteredOptions is set
6. Slice is used to remove the array that is passed inside it as an argument from the source array
7. Where filter() is used to filter and return a new array which is passed inside it from the source array
8. indexOf 🡺 returns position of the passed element and its start from 0 for the First position of the element
9. here Custom Pipe is used but needed more information on the second input paramenter of the Pipe “async” (See the red color underlined line in html part code
10. The output for the above code is 🡺

Initially🡺

On Entering First character 🡺

Now let’s see small change in filter method and see how output behaves🡺 🡺

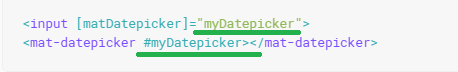
initially indexOf was compared with 0 now let’s see the output

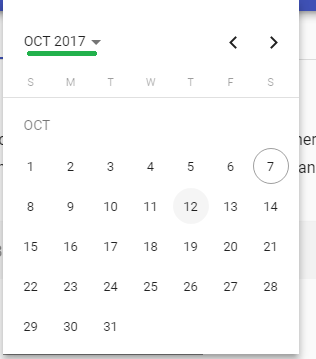
** 🡺**

**this is because of IndexOf Value**

1. **Checkbox**
2. **<mat-checkbox>** provides the same functionality as a native <input type="checkbox"> enhanced with Material Design styling and animations.
3. 

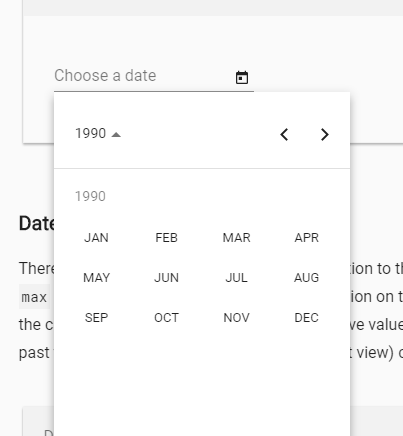
### Checkbox label

1. The checkbox label is provided as the content to the <mat-checkbox> element. The label can be positioned before or after the checkbox by setting the labelPosition property to 'before' or 'after'.
2. If you don't want the label to appear next to the checkbox, you can use [aria-label](https://www.w3.org/TR/wai-aria/states_and_properties#aria-label) or [aria-labelledby](https://www.w3.org/TR/wai-aria/states_and_properties#aria-labelledby)to specify an appropriate label.
3. The color of a <mat-checkbox> can be changed by using the color property. By default, checkboxes use the theme's accent color. This can be changed to 'primary' or 'warn'.
4. **Datepicker**
5. A datepicker is composed of a text input and a calendar pop-up, connected via the matDatepicker property on the text input.
6. 
7. <mat-datepicker-toggle> 🡺 we will see this later with an example
8. By default the calendar will open in month view, as shown below

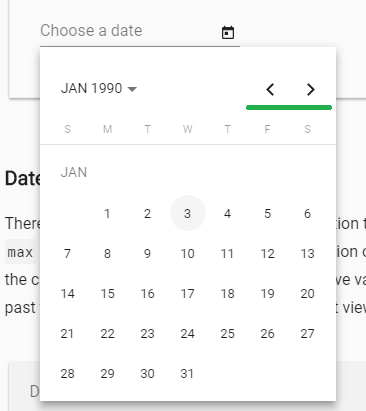


1. this can be changed by setting the startView property of mat-datepicker to "year". In year view the user will see all months of the year and then proceed to month view after choosing a month. As shown below

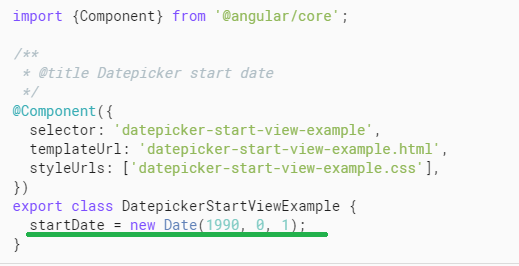
part 1🡺



Part2 🡺 on clicking Jan



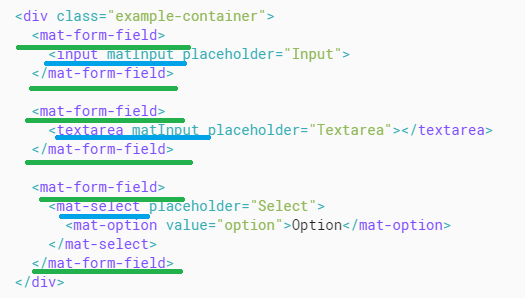




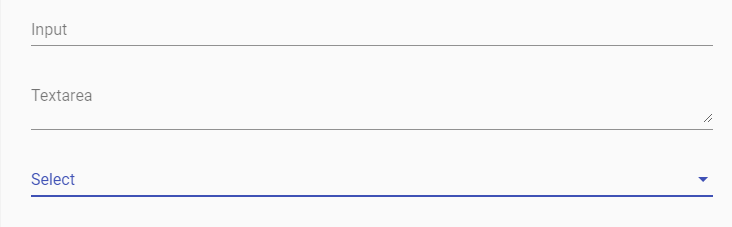
### Date validation 🡺 later

### Form field

1. <mat-form-field> 🡺 refers to the wrapper component
2. The following Angular Material components are designed to work inside a <mat-form-field>:
3. [<input matInput> & <textarea matInput>](https://material.angular.io/components/input/overview)
4. <mat-select>
5. <mat-chip-list>
6. Example🡺



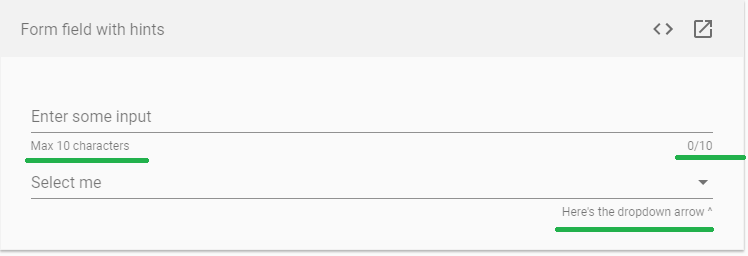
Output:🡺



1. **Floating placeholder** 🡺
2. The floating placeholder is a text label displayed on top of the form field control when the control does not contain any text
3. ***By default, when text is present the floating placeholder floats above the form field control.***
4. Placeholder text can be specified using the placeholder property on the form field control, or by adding a <mat-placeholder> element inside the form field. Only one of these options should be used, specifying both will raise an error.
5. If the form field control is marked with a required attribute, an asterisk will be appended to the placeholder to indicate the fact that it is a required field. If unwanted, this can be disabled by setting the **hideRequiredMarker** property on <mat-form-field>
6. The**floatPlaceholder** property of <mat-form-field> can be used to change this default floating behavior. It can set to **never** to hide the placeholder instead of float it when text is present in the form field control. It can be set to **always** to float the placeholder even when no text is present in the form field control. It can also be set to **auto** to restore the default behavior.
7. I was not able to see the code in doc some error is displaying

### Hint labels

1. Hint labels are additional descriptive text that appears below the form field's underline
2. A <mat-form-field> can have up to two hint labels; one start-aligned (left in an LTR language, right in RTL), and one end-aligned.
3. Hint labels are specified in one of two ways: either by using the **hintLabel** property of <mat-form-field>, or by adding a <**mat**-**hint**> element inside the form field
4. When adding a hint via the hintLabel property, it will be treated as the start hint
5. Hints added via the <mat-hint> hint element can be added to either side by setting the align property on <mat-hint> to either start or end. Attempting to add multiple hints to the same side will raise an error.



### Error messages

### Error messages can be shown under the form field underline by adding mat-error elements inside the form field

### Errors are hidden initially and will be displayed on invalid form fields after the user has interacted with the element or the parent form has been submitted. Since the errors occupy the same space as the hints, the hints are hidden when the errors are shown.

### If a form field can have more than one error state, it is up to the consumer to toggle which messages should be displayed. This can be done with CSS, ngIf or ngSwitch.

### 

### Code🡺 error msg displayed

### Prefix & suffix

### Custom content can be included before and after the input tag, as a prefix or suffix by using adding the matPrefix and matSuffix directive to an element inside the <mat-form-field>

### 

### 

### 

### Custom form field controls and Theming 🡺 later

### Input

### Supported input types

### The following [input types](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input) can be used with matInput

* date
* datetime-local
* email
* month
* number
* password
* search
* tel
* text
* time
* url
* week

### Radio button

### mat-radio> provides the same functionality as a native <input type="radio"> enhanced with Material Design styling and animations.

### 

### 

### Radio-button label🡺 The label can be positioned before or after the radio-button by setting the labelPosition property to 'before' or 'after'.

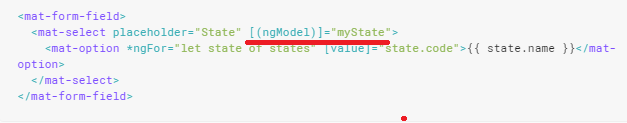
### Radio groups

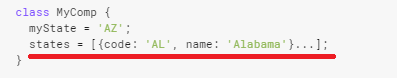
### Radio-buttons should typically be placed inside of an <mat-radio-group>

### The radio-group has a valueproperty that reflects the currently selected radio-button inside of the group

# **Select**

* + 1. <mat-select> is a form control for selecting a value from a set of options, similar to the native<select> element
    2. Note that you can disable items by adding the **disabled** Boolean attribute or binding to it.
    3. 
    4. 2nd example by using 2 way data binding



* + 1. 

### Resetting the select value

* + 1. Example🡺
    2. Output 🡺
       1. Initially 🡺
       2. On Placing the cursor in the select field🡺
       3. On selecting “None” in the option🡺

### Customizing the trigger label

* + 1. **If you want to display a custom trigger label inside a select, you can use the mat-select-trigger element 🡺 Example later**

# **Slider**

1. **<mat-slider>** allows for the selection of a value from a range via mouse, touch, or keyboard, similar **to <input type="range">.**
2. 
3. By default the minimum value of the slider is 0, the maximum value is 100, and the thumb moves in increments of 1. **These values can be changed by setting the min, max, and step attributes respectively**. The initial value is set to the minimum value unless otherwise specified.
4. 

### Orientation

1. By default sliders are horizontal with the minimum value on the left and the maximum value on the right. The vertical attribute can be added to a slider to make it vertical with the minimum value on bottom and the maximum value on top.
2.  🡺 output🡺 

### 🡺

### Output🡺

# **Slide toggle**

### <mat-slide-toggle> is an on/off control that can be toggled via clicking or dragging.

### 

### 

# **Menu**

### <mat-menu> is a floating panel containing list of options.

### Example 🡺

### 

### On Clicking Menu button (see point b)🡺

### Nested menu 🡺 now lets see the syntax for nested menu

### 

### On clicking the above Animal Index button 🡺

### Code🡺

### 

1. **Needed More information on 🡺@ViewChild**

### Now lets see an example with ICON and text in the menu

### 

### Code🡺and Final Output for the above code

### 

### Customizing menu position

### By default, the menu will display below (y-axis), after (x-axis),

### The position can be changed using the xPosition (before | after) and yPosition (above | below) attributes.

### 

### Output 🡺

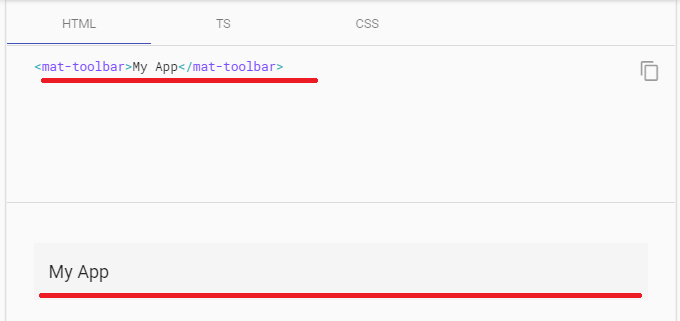
### on clicking the vert icon 🡺

### Nested menu example for above scenario🡺 code we will see later now lets see just the output

### 

# **Sidenav 🡺 Later**

# **Toolbar**

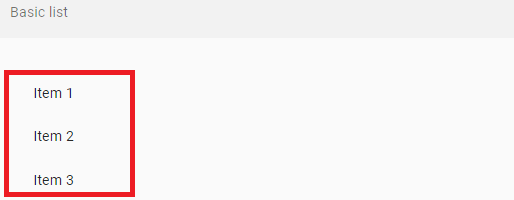
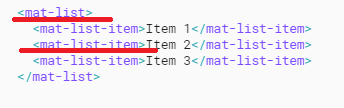
1. <mat-toolbar> is a container for headers, titles, or actions.

**Layouts**

# **List**

### Type of list🡺 simple list, Navigation list, Selection lists, Multi-line lists, Lists with icons, Lists with avatars(means images), Dense lists, Lists with multiple sections

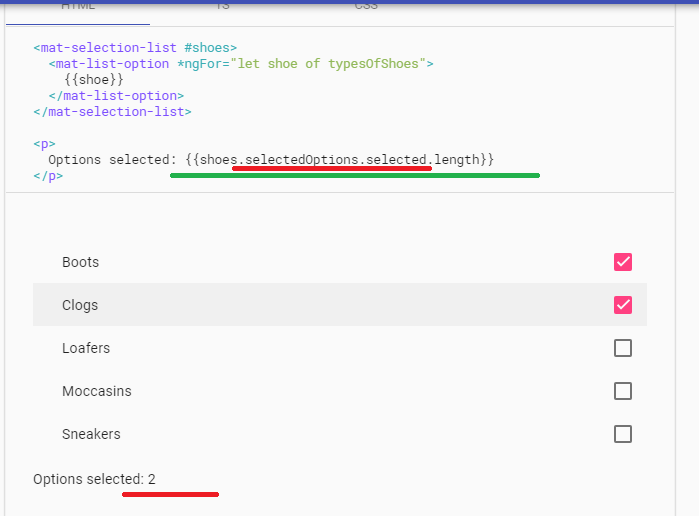
,

1. <mat-list> is a container component that wraps and formats a series of line items.
2. 
3. Code for Simple list🡺

### Navigation lists

1. Use mat-nav-list tags for navigation lists (i.e. lists that have anchor tags).
2. 
3. 

### Selection lists

1. 



### Dense lists

1. Lists are also available in "dense layout" mode, which shrinks the font size and height of the list to suit UIs that may need to display more information. To enable this mode, add a dense attribute to the main mat-list tag.

# **Grid list**

### mat-grid-list is a two-dimensional list view that arranges cells into grid-based layout.

### 

### Output🡺

### Setting the number of columns🡺 ms for a grid list

### An mat-grid-list must specify a cols attribute which sets the number of columns in the grid. The number of rows will be automatically determined based on the number of columns and the number of items.

### Setting the row height 🡺 ms for a grid list

### 

### mat-grid-tile 🡺 A header and footer can be added to an mat-grid-tile using the mat-grid-tile-header and mat-grid-tile-footer elements respectively.

# **Card**

### <mat-card> is a content container for text, photos, and actions in the context of a single subject.

### 

### 

### Basic card sections🡺some of the attributes and elements that can be added in the card is🡺

# **Stepper**

### Angular Material's stepper provides a wizard-like workflow by dividing content into logical steps.

### Example 🡺

### Step1🡺

### 

### Step2🡺

### Step3🡺

### 

“Enable linear mode” 🡺 this button is used to enable the form validation that is provided by material on clicking on it

### 

### 

### Stepper variants 🡺There are two stepper components: mat-horizontal-stepper and mat-vertical-stepper. They can be used the same way. The only difference is the orientation of stepper.

### Labels 🡺 if a step's label is only text, then the label attribute can be used🡺

### For more complex labels, add a template with the matStepLabel directive inside the mat-step.

### Stepper buttons🡺 There are two button directives to support navigation between different steps: matStepperPrevious and matStepperNext.

### 

### Linear stepper🡺 The linear attribute can be set on mat-horizontal-stepper and mat-vertical-stepper to create a linear stepper that requires the user to complete previous steps before proceeding to following steps.

# **Tabs**

### 

### dynamicHeight  🡺 Needed more information on this attribute of tab section

### Tabs and navigation

### 

# **Expansion Panel**

### <mat-expansion-panel> provides an expandable details-summary view.

### Output 🡺

### Now on clicking Down ward arrow

### 

### Code🡺

#### Action bar🡺 Actions may optionally be included at the bottom of the panel, visible only when the expansion is in its expanded state.

#### **Disabling a panel** 🡺

### Accordion🡺 Multiple expansion-panels can be combined into an accordion. *The multi="true" input allows the expansions state to be set independently of each other. When multi="false" (default) just one panel can be expanded at a given time:*

### Example 🡺

### Now on clicking the down ward arrow we get the following output

### 

### <mat-accordion multi="true"> 🡺 allows us to open all the expandable sections independent of each other

**BUTTONS & INDICATORS**

# **Button**

### Angular Material buttons are native <button> or <a> elements enhanced with Material Design styling and ink ripples.

### 🡺

### 

# **Button toggle**

### a. <mat-button-toggle> are on/off toggles with the appearance of a button.

<mat-button-toggle>Toggle me!</mat-button-toggle> 🡺

Output:



Now on clicking the above button

### 

# **Chips**

### <mat-chip-list> displays a list of values as individual, keyboard accessible, chips.

### 

### 

### Unstyled chips 🡺 By default, <mat-chip> has Material Design styles applied.

1. Selection🡺 Chips can be selected via the selected property. Selection can be disabled by setting selectable to false on the <mat-chip-list>.Whenever the selection state changes, a **ChipSelectionChange** event will be emitted via(**selectionChange**).

### Disabled chips🡺 Individual chips may be disabled by applying the disabled attribute to the chip. When disabled, chips are neither selectable nor focusable

### Orientation 🡺 If you want the chips in the list to be stacked vertically, instead of horizontally, you can apply the mat-chip-list-stacked class, as well as the aria-orientation="vertical" attribute:

### 

# **Icon**

### mat-icon makes it easier to use *vector-based* icons in your app. This directive supports both icon fonts and SVG icons,

### 🡺

### Remaining topics under this section we will see it later

# **Progress spinner**

### <mat-progress-spinner> and <mat-spinner> are a circular indicators of progress and activity.

### 

### 

### Note 🡺 Progress Snipper is circular loading symbol and Progress bar is horizontal line to show the progress

### 

### Or

### 

# **Progress bar**

### <mat-progress-bar> is a horizontal progress-bar for indicating progress and activity.

### 

### 

### Now lets see the example for all the above modes🡺

### Determinate🡺

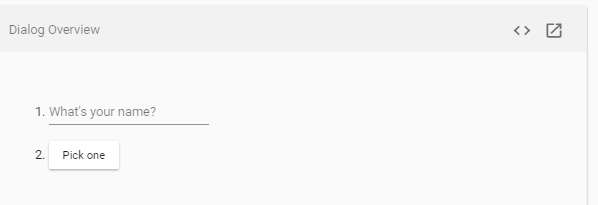
### Indeterminate🡺

### Buffer🡺

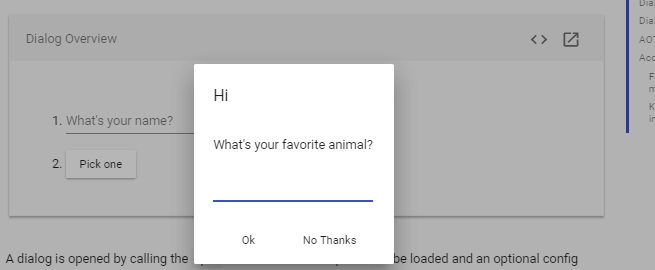
### Query🡺 is just moves in the opposite direction of Indeterminate

### POPUPS & MODALS

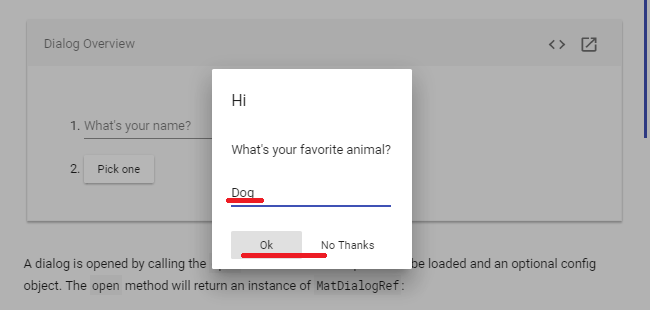
# **Dialog**

1. The **MatDialog** service can be used to open modal dialogs with Material Design styling and animations.
2. 

Now on click ok “Pick One” button, we get the following dialog box

 🡺 see here apart from the modal all other contents are disabled

Now Type “Dog” and click on Ok button as shown below



Now On click of ok we get the following output

### 

### Code 🡺

### @Inject 🡺 needed to know the purpose of This Annotation

### Step 1🡺

### Step 2🡺 Here let’s see the content of the DialogOverviewExample Component🡺 needed more explanation on why Subscribe() method is used here

### And for what is the “data” object used for here

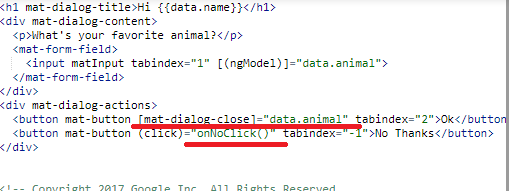
### 

**Step 3🡺 Here let’s see the content of the** DialogOverviewExampleDialog

Doubts🡺 here constructor part and onNoClick() method is not clear so I need more info on this

### 

**Step 4🡺now the content of the “**dialog-overview-example-dialog.html”



Here needed more information on the purpose of “mat-dialog-close” 🡺 note on closing the dialop box what is the further action needed to be taken is explained by

“dialogRef.afterClosed().subscribe(-----)”

### Note 🡺 for detailed explanation we can come back to this documentation again

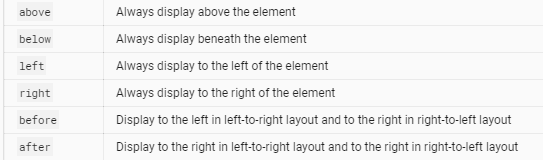
# **Tooltip**

### The Angular Material tooltip provides a text label that is displayed when the user hovers over or long presses an element.

### Code🡺

### Output🡺

1. **Positioning🡺**The tooltip will be displayed below the element but this can be configured using the **matTooltipPosition** input. The tooltip can be displayed above, below, left, or right of the element. By default the position will be below.

****

### Showing and hiding🡺 The tooltip is immediately shown when the user's mouse hovers over the element and immediately hides when the user's mouse leaves. A delay in showing or hiding the tooltip can be added through the inputs matTooltipShowDelay and matTooltipHideDelay.

# **Snackbar**

### MatSnackBar is a service for displaying snack-bar notifications.

### Lets see an example on Snackbar 🡺

### 

### Now let’s click on “Show snack-bar” and we get the snack bar as shown below🡺

### 

### Code🡺first let’s see the html content

### 

### Now let’s see the component file contents

### 

### Needed more information on the Syntax of the open()

# **Table**

### The mat-table provides a Material Design styled data-table that can be used to display rows of data.

### This table builds on the foundation of the CDK data-table and uses a similar interface for its data source input and template, except that its element and attribute selectors will be prefixed with mat- instead of cdk-.

### RXJS Library

# [**ReactiveX**](https://github.com/ReactiveX)**/**[rxjs](https://github.com/ReactiveX/rxjs)**🡺**A **reactive programming library** for JavaScript

#### Operators🡺[RxJS 5 operators](https://www.learnrxjs.io/operators/)

### A complete list of RxJS 5 operators,

### [Combination](https://www.learnrxjs.io/operators/combination/)

### [Conditional](https://www.learnrxjs.io/operators/conditional/)

### [Creation](https://www.learnrxjs.io/operators/creation/)

### [Error Handling](https://www.learnrxjs.io/operators/error_handling/)

### [Filtering](https://www.learnrxjs.io/operators/filtering/)

### [Multicasting](https://www.learnrxjs.io/operators/multicasting/)

### [Transformation](https://www.learnrxjs.io/operators/transformation/)

### [Utility](https://www.learnrxjs.io/operators/utility/)

### Now let’s see the operators in detail

### [Combination](https://www.learnrxjs.io/operators/combination/)

### The combination operators allow the joining of information from multiple observables.

### Order, time, and structure of emitted values is the primary variation among these operators.

### The List of Combination Operators are

* [combineAll](https://www.learnrxjs.io/operators/combination/combineall.html)
* [**combineLatest**](https://www.learnrxjs.io/operators/combination/combinelatest.html)
* [**concat**](https://www.learnrxjs.io/operators/combination/concat.html)
* [concatAll](https://www.learnrxjs.io/operators/combination/concatall.html)
* [forkJoin](https://www.learnrxjs.io/operators/combination/forkjoin.html)
* [**merge**](https://www.learnrxjs.io/operators/combination/merge.html)
* [mergeAll](https://www.learnrxjs.io/operators/combination/mergeall.html)
* [pairwise](https://www.learnrxjs.io/operators/combination/pairwise.html)
* [race](https://www.learnrxjs.io/operators/combination/race.html)
* [**startWith**](https://www.learnrxjs.io/operators/combination/startwith.html)
* [**withLatestFrom**](https://www.learnrxjs.io/operators/combination/withlatestfrom.html)
* [zip](https://www.learnrxjs.io/operators/combination/zip.html)

### combineAll 🡺

### Takes an Observable of Observables, and collects all Observables from it. Once the outer Observable completes, it subscribes to all collected Observables and combines their values using the [combineLatest](http://reactivex.io/rxjs/class/es6/Observable.js~Observable.html#static-method-combineLatest) strategy, such that:

### Syntax🡺public combineAll(project: [function](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Function)): [Observable](http://reactivex.io/rxjs/class/es6/Observable.js~Observable.html)

### 

### Example 🡺 this observable is making use of other observables so Later we will see the example

### combineLatesr 🡺

### Combines multiple Observables to create an Observable whose values are calculated from the latest values of each of its input Observables.

### 

### Syntax Explanation🡺

### Example🡺 here “of” operator has used so needed to know this purpose first

### Concat🡺Creates an output Observable which sequentially emits all values from every given input Observable after the current Observable.

### 

### Syntax🡺

### startWith🡺Returns an Observable that emits the items you specify as arguments before it begins to emit items emitted by the source Observable.

### debounceTime🡺Emits a value from the source Observable only after a particular time span has passed without another source emission.

### Debounce🡺

### distinctUntilChanged🡺Only emit when the current value is different than the last.

### Filter🡺Emit values that pass the provided condition.

### Throw🡺Creates an Observable that emits no items to the Observer and immediately emits an error notification.

### Catch🡺Gracefully handle errors in an observable sequence.

### Retry🡺Returns an Observable that mirrors the source Observable with the exception of an error. If the source Observable calls error, this method will resubscribe to the source Observable for a maximum of count resubscriptions (given as a number parameter) rather than propagating the error call. 🡺

### retryWhen🡺Retry an observable sequence on error based on custom criteria 🡺

### fromPromise🡺Create observable from promise, emitting result.

### toPromise 🡺 Convert observable to promise🡺

### 

### Map🡺You use map to transform a collection of items into a collection of different items

### mapTo🡺

### switch🡺 MS🡺Observe following diagram carefully this is what Switch Operator of RXJS does🡺  ms that as soon as the next trigger emits, the current async operation is canceled and retriggered.

### SwitchMap🡺

### concatMap🡺

### flatMap🡺

### 