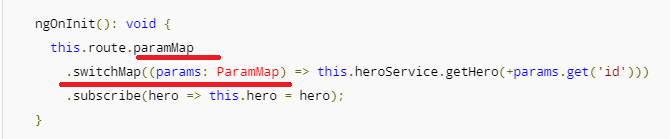
1. Although [NgModel](https://angular.io/api/forms/NgModel) is a valid Angular directive, it isn't available by default. It belongs to the optional [FormsModule](https://angular.io/api/forms/FormsModule). You must opt-in to using that module.
2. Immediately I save the TS file corresponding JS and js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

Export class ClassName{

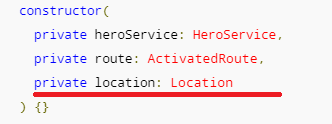
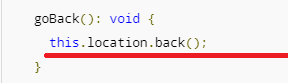
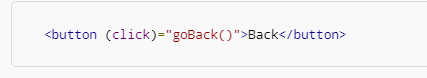
}

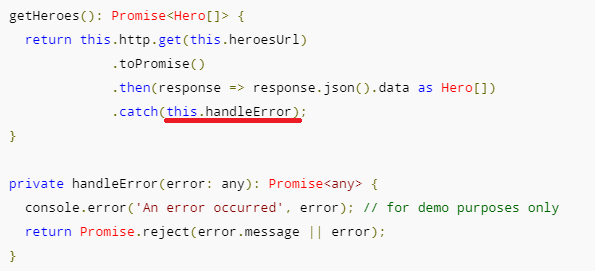
1. An simple example on how to compare in Angular 2is 🡺\*ngIf = ”obj1===obj2”
2. import { [Component](https://angular.io/api/core/Component), [Input](https://angular.io/api/core/Input) } from '@angular/core';
3. **Examples on @Input we will see this once after seeing DPK notes**
4. While Using the Promise I got the Following error in the console🡺

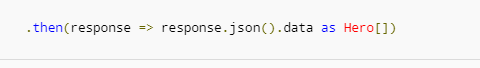
**“Cannot find a differ supporting object ‘[object Promise]’ of type ‘object’ . NgFor only supports binding to iterables such as arrays”.**

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

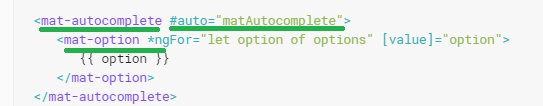
“[ts] class InMemoryDataService\_ MyCUSTOMCLass incompletely implements interface InMemoryDbService property cretaeDB is missing in type InMemoryDataService\_MyCUSTOMCLass”

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**

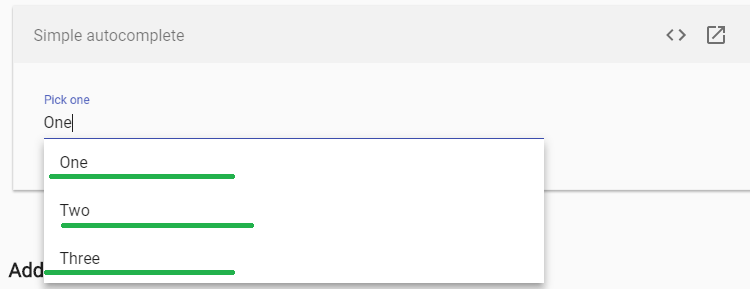
1. **Autocomplete**
2. The autocomplete is a normal input text box enhanced by a panel of suggested options
3. First lets see how to create an angular based input field



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🡺

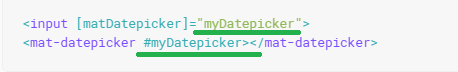
Part2:While placing the Cursor in the text box

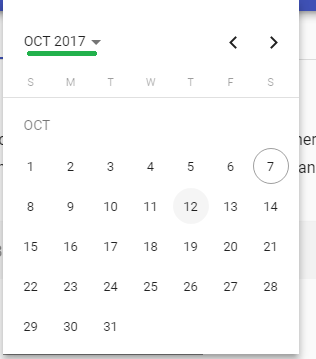


### Part2 of AutoComplete 🡺 Adding a custom filter

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
7. **Checkbox**
8. **<mat-checkbox>** provides the same functionality as a native <input type="checkbox"> enhanced with Material Design styling and animations.
9. 

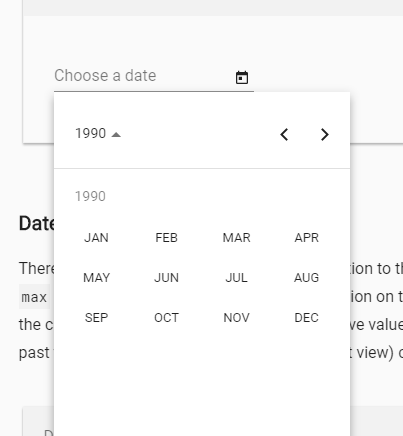
### Checkbox label

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

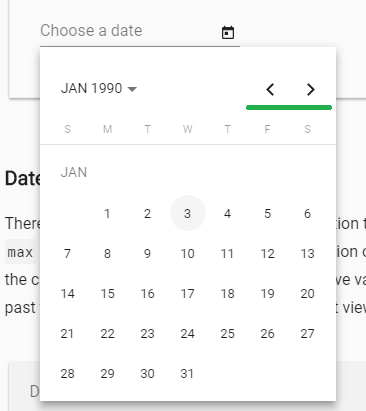


1. this can be changed by setting the startView property of mat-datepicker to "year". In year view the user will see all months of the year and then proceed to month view after choosing a month. As shown below

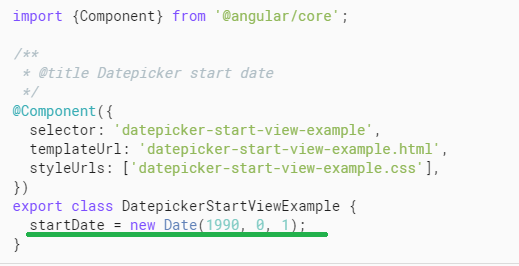
part 1🡺



Part2 🡺 on clicking Jan



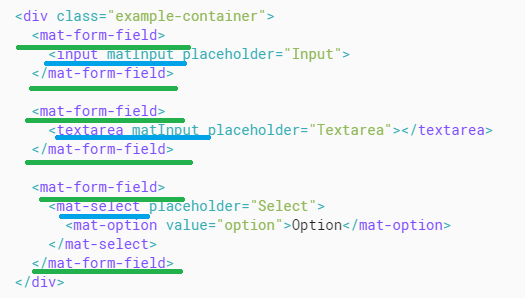




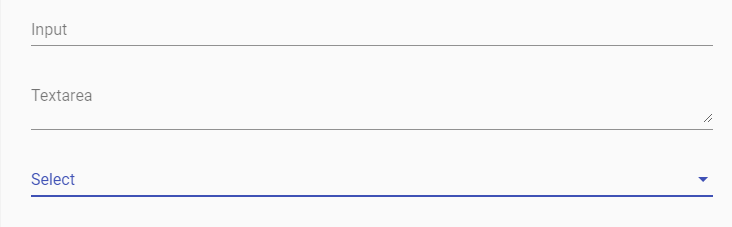
### Date validation 🡺 later

### 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🡺



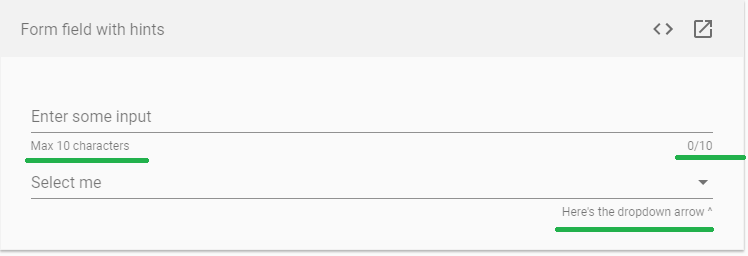
Output:🡺



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'.