FREE eBook

LEARNING angular-material2

Free unaffiliated eBook created from
Stack Overflow contributors.

#angular-material2

Table of Contents

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: angular-material2

It is an unofficial and free angular-material2 ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official angular-material2.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with angular-material2

Remarks

This section provides an overview of what angular-material2 is, and why a developer might want to use it.

It should also mention any large subjects within angular-material2, and link out to the related topics. Since the Documentation for angular-material2 is new, you may need to create initial versions of those related topics.

Versions

Version	Changelog	Date
2.0.0-beta.8	Link	2017-07-06
2.0.0-beta.7	Link	2017-06-19
2.0.0-beta.6	Link	2017-05-25
2.0.0-beta.5	Link	2017-05-13
2.0.0-beta.4	Link	2017-05-12
2.0.0-beta.3	Link	2017-04-07
2.0.0-beta.2	Link	2017-02-15
2.0.0-beta.1	Link	2016-12-23
2.0.0-beta.0	Link	2016-12-22

Examples

Installation or Setup with Angular CLI

In this example, we will be using `@angular/cli` (latest) and the latest version of `@angular/material`. You should at least know the basics of Angular 2/4 before continuing the steps below.

1. Install angular material module from `npm`:

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

2.0.0-beta.3

This only applies to versions `2.0.0-beta.3` and up.

Install the `@angular/animations` module:

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

2.0.0-beta.8

This only applies to versions `2.0.0-beta.8` and up.

Install the `@angular/cdk` module:

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

2. In your application module import the components which you are going to use:

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { RouterModule } from '@angular/router';
import { MdButtonModule, MdSnackBarModule, MdSidenavModule } from '@angular/material';

import { AppComponent } from './app.component';

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

@NgModule({
     imports: [
         BrowserAnimationsModule,
         MdButtonModule,
         MdSnackBarModule,
         MdSidenavModule,
         CommonModule,
         // This is optional unless you want to have routing in your app
         // RouterModule.forRoot([
         //     { path: '', component: HomeView, pathMatch: 'full'}
         // ])
     ],
     declarations: [ AppComponent ],
     boostrap: [ AppComponent ]
})
export class AppModule {}
```

You are now ready to use Angular Material in your components!

Note: The docs for specific components will be coming soon.

Wrapping all modules together

You can also easily wrap all angular modules, which you are going to use, into one module:

```
import { NgModule } from '@angular/core';
import { MdButtonModule, MdSnackBarModule, MdSidenavModule } from '@angular/material';
```

```
@NgModule({
    imports: [
        BrowserAnimationsModule,
        MdButtonModule,
        MdSnackBarModule,
        MdSidenavModule
    ],
    exports: [
        MdButtonModule,
        MdSnackBarModule,
        MdSidenavModule
    ]
})
export class MaterialWrapperModule {}
```

After that simply import your module into the application main module:

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { RouterModule } from '@angular/router';

import { MaterialWrapperModule } from './material-wrapper.module.ts';
import { AppComponent } from './app.component';


@NgModule({
    imports: [
        BrowserAnimationsModule,
        MaterialWrapperModule,
        CommonModule,
        // This is optional, use it when you would like routing in your app
        // RouterModule.forRoot([
        //     { path: '', component: HomeView, pathMatch: 'full'}
        // ])
    ],
    declarations: [ AppComponent],
    bootstrap: [ AppComponent ]
})
export class AppModule {}
```

Installation and Setup from master with Angular CLI

This example will be how to install from master and will be using `@angular/cli` as well.

1. Make a new project with `@angular/cli`:

   ```
   ng new my-master-app
   ```

   If you haven't installed `@angular/cli`, use this command:

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

2. Install from `master`:

```
npm install --save @angular/animations
npm install --save angular/material2-builds
npm install --save angular/cdk-builds
```

3. Follow the same guide as above.

Done!

Set up theme, gesture support and material icons

Theme:

*A theme is **required** for material components to work properly within the application.*

Angular Material 2 provides four prebuilt themes:

- deeppurple-amber
- indigo-pink
- pink-bluegrey
- purple-green

If you are using **Angular CLI**, you can import one of the prebuilt themes in `style.css`.

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

Theme can be added using `<link>` in `index.html` as well.

```
<link href="node_modules/@angular/material/prebuilt-themes/indigo-pink.css" rel="stylesheet">
```

HammerJS

Add HammerJS to the application via CDN or `npm`:

```
npm install --save hammerjs
```

In your root module, usually `app.module.ts`, add the import statement:

```
import 'hammerjs';
```

Material Icons:

Unless, custom icons provided, Angular Material 2 `<md-icon>` expects Material Icons.

In `index.html` add:

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

Read Getting started with angular-material2 online: https://riptutorial.com/angular-material2/topic/10828/getting-started-with-angular-material2

Chapter 2: md-autocomplete

Introduction

This topic includes coding examples related to Angular Material 2 Autocomplete (md-autocomplete)

Remarks

These examples don't cover all features of md-autocomplete. Please read the documentation learn more about md-autocomplete.

Examples

Separate control and display

This example shows how to to display specific property in dropdown but bind with the whole object.

autocomplete-overview-example.html:

```
<md-input-container>
  <input mdInput placeholder="State" [(ngModel)]="selection"
         [mdAutocomplete]="auto" [formControl]="stateCtrl">
</md-input-container>

<md-autocomplete #auto="mdAutocomplete" [displayWith]="displayFn">
  <md-option *ngFor="let state of filteredStates | async" [value]="state" >
    {{ state.Country }}
  </md-option>
</md-autocomplete>


<p>Selected Country: {{selection | json}}</p>
<p>Selected Country Id: {{selection?.CountryID}}</p>
```

autocomplete-overview-example.ts:

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

import 'rxjs/add/operator/startWith';
import 'rxjs/add/operator/map';

@Component({
  selector: 'autocomplete-overview-example',
  templateUrl: 'autocomplete-overview-example.html',
})
export class AutocompleteOverviewExample {
  stateCtrl: FormControl;
```

```
  filteredStates: any;

  selection: any;

  states = [
    { Country: "United States Of America" , CountryID: "1"},
    { Country: "United Kingdom" , CountryID: "2"},
    { Country: "United Arab Emirates" , CountryID: "3"},
  ];

  constructor() {
    this.stateCtrl = new FormControl();
    this.filteredStates = this.stateCtrl.valueChanges
        .startWith(null)
        .map(country => country && typeof country === 'object' ? country.Country : country)
        .map(name => this.filterStates(name));
  }

  filterStates(val) {
    return val ? this.states.filter(s => s.Country.toLowerCase().indexOf(val.toLowerCase())
=== 0)
               : this.states;
  }

  displayFn(country): string {
    console.log(country);
      return country ? country.Country : country;
   }

}
```

Live Example

Get md-autocomplete's options/searchable data from API

data.service.ts:

```
import { Injectable } from '@angular/core';
import { Http } from '@angular/http';
import 'rxjs/add/operator/map';

@Injectable()
export class DataService {

  constructor(private http: Http) { }

  fetchData(){
    return this.http.get('https://dinstruct-d4b62.firebaseio.com/.json')
      .map((res) => res.json())

  }

}
```

autocomplete-overview-example.html:

```
<md-input-container>
```

```
  <input mdInput placeholder="Name" [mdAutocomplete]="auto" [formControl]="stateCtrl">
</md-input-container>

<md-autocomplete #auto="mdAutocomplete" [displayWith]="displayFn">
  <md-option *ngFor="let sector of filteredSectors | async" [value]="sector">
    {{ sector.name }}
  </md-option>
</md-autocomplete>

<div>
    <h2>Data :</h2>
    <span>{{ allSectors | json  }}</span>
</div>
```

autocomplete-overview-example.ts:

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

import { DataService } from './data.service';
import 'rxjs/add/operator/startWith';
import 'rxjs/add/operator/map';

@Component({
  selector: 'autocomplete-overview-example',
  templateUrl: './autocomplete-overview-example.html',
})
export class AutocompleteOverviewExample implements OnInit{
  stateCtrl: FormControl;

  filteredSectors: any;

  allSectors;

  constructor(private dataService: DataService) {
    this.stateCtrl = new FormControl();
  }

  ngOnInit(){
    this.dataService.fetchData()
      .subscribe(
        (data) => {
          this.allSectors = data.customers;
          this.filteredSectors = this.stateCtrl.valueChanges
            .startWith(null)
            .map(val => val ? this.filter(val) : this.allSectors.slice());
        }
    );

  }

  filter(name) {
   return this.allSectors.filter(sector => new RegExp(`^${name}`, 'gi').test(sector.name));
  }

   displayFn(sector) {
      return sector ? sector.name : sector;
   }

}
```

Live Example

Utilize md-autocomplete inside a reactive form

This example requires `FormsModule` and `ReactiveFormsModule`. Please import them in your application/module.

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

input-form-example.html

```
<form class="example-form" (ngSubmit)="submit(addForm.value)" [formGroup]="addForm">
  <md-input-container class="example-full-width">
    <input mdInput placeholder="Company (disabled)" disabled value="Google"
formControlName="company">
  </md-input-container>

  <table class="example-full-width" cellspacing="0"><tr>
    <td><md-input-container class="example-full-width">
      <input mdInput placeholder="First name" formControlName="fname">
    </md-input-container></td>
    <td><md-input-container class="example-full-width">
      <input mdInput placeholder="Long Last Name That Will Be Truncated">
    </md-input-container></td>
  </tr></table>

  <p>
    <md-input-container class="example-full-width">
      <textarea mdInput placeholder="Address" formControlName="address">1600 Amphitheatre
Pkwy</textarea>
    </md-input-container>
    <md-input-container class="example-full-width">
      <textarea mdInput placeholder="Address 2"></textarea>
    </md-input-container>
  </p>

  <table class="example-full-width" cellspacing="0"><tr>
    <td><md-input-container class="example-full-width">
      <input mdInput placeholder="City" formControlName="city">
    </md-input-container></td>

    <td><md-input-container>
      <input mdInput placeholder="State"
            [mdAutocomplete]="auto"
            [formControl]="stateCtrl"
            formControlName="state">
    </md-input-container></td>

    <td><md-input-container class="example-full-width">
      <input mdInput #postalCode maxlength="5" placeholder="Postal Code" value="94043"
formControlName="zip">
      <md-hint align="end">{{postalCode.value.length}} / 5</md-hint>
    </md-input-container></td>
  </tr></table>

   <button md-raised-button  type="submit">Submit</button>

   <md-autocomplete #auto="mdAutocomplete" >
```

```
    <md-option *ngFor="let state of filteredStates | async" [value]="state"
(onSelectionChange)="selectState(state, addForm.value)">
      {{ state }}
    </md-option>
  </md-autocomplete>

</form>

<p>Form values:</p>
<p>{{ addForm.value | json }}</p>
```

input-form-example.ts:

```
import {Component} from '@angular/core';
import {FormBuilder, FormGroup, FormControl} from '@angular/forms';
import 'rxjs/add/operator/startWith';
import 'rxjs/add/operator/map';

@Component({
  selector: 'input-form-example',
  templateUrl: 'input-form-example.html',
  styleUrls: ['input-form-example.css'],
})
export class InputFormExample {
  stateCtrl: FormControl;
  filteredStates: any;

  addForm: FormGroup;

  state;

  states = [
    'Alabama',
    'Alaska',
    'Arizona',
    'Arkansas',
    'California',
    'Colorado',
    'Connecticut',
    'Delaware',
    'Florida',
    'Georgia',
    'Hawaii',
    'Idaho',
    'Illinois',
    'Indiana',
    'Iowa',
    'Kansas',
    'Kentucky',
    'Louisiana',
    'Maine',
    'Maryland',
    'Massachusetts',
    'Michigan',
    'Minnesota',
    'Mississippi',
    'Missouri',
    'Montana',
    'Nebraska',
    'Nevada',
```

```
    'New Hampshire',
    'New Jersey',
    'New Mexico',
    'New York',
    'North Carolina',
    'North Dakota',
    'Ohio',
    'Oklahoma',
    'Oregon',
    'Pennsylvania',
    'Rhode Island',
    'South Carolina',
    'South Dakota',
    'Tennessee',
    'Texas',
    'Utah',
    'Vermont',
    'Virginia',
    'Washington',
    'West Virginia',
    'Wisconsin',
    'Wyoming',
  ];

  constructor(private fb: FormBuilder) {

    this.addForm = this.fb.group({
      fname: '',
      address: '',
      address2: '',
      city: '',
      "state": this.state,
      zip: '',
      company: '',
      lname: ''
    });
    this.stateCtrl = new FormControl();
    this.filteredStates = this.stateCtrl.valueChanges
        .startWith(null)
        .map(name => this.filterStates(name));
  }

  filterStates(val: string) {
    return val ? this.states.filter(s => new RegExp(`^${val}`, 'gi').test(s))
               : this.states;
  }

  submit(form){
    alert(JSON.stringify(form));
  }

  selectState(state, form){
    // console.log(state);
    // console.log(form);
    form.state = state;
  }
}
```

input-form-example.css:

```
.example-form {
  width: 500px;
}

.example-full-width {
  width: 100%;
}
```

Live Example

One md-autocomplete for multiple formControl

This example requires `FormsModule` and `ReactiveFormsModule`. Please import them in your application/module.

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

autocomplete-overview-example.html:

```
<md-input-container>
  <input mdInput placeholder="State" [mdAutocomplete]="auto" [formControl]="stateCtrl">
</md-input-container>

<p></p>

<md-input-container>
  <input mdInput placeholder="State2" [mdAutocomplete]="auto" [formControl]="stateCtrl2">
</md-input-container>

<p></p>

<md-input-container>
  <input mdInput placeholder="State3" [mdAutocomplete]="auto" [formControl]="stateCtrl3">
</md-input-container>

<md-autocomplete #auto="mdAutocomplete">
  <md-option *ngFor="let state of filteredStates | async" [value]="state">
    {{ state }}
  </md-option>
</md-autocomplete>
```

autocomplete-overview-example.ts:

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

import 'rxjs/add/operator/startWith';
import 'rxjs/add/operator/map';

@Component({
  selector: 'autocomplete-overview-example',
  templateUrl: 'autocomplete-overview-example.html',
})
export class AutocompleteOverviewExample {
  stateCtrl: FormControl;
  stateCtrl2: FormControl;
```

```
  stateCtrl3: FormControl;
  filteredStates: any;

  states = [
    'Alabama',
    'Alaska',
    'Arizona',
    'Arkansas',
    'California',
    'Colorado',
    'Connecticut',
    'Delaware',
    'Florida',
    'Georgia',
    'Hawaii',
    'Idaho',
    'Illinois',
    'Indiana',
    'Iowa',
    'Kansas',
    'Kentucky',
    'Louisiana',
    'Maine',
    'Maryland',
    'Massachusetts',
    'Michigan',
    'Minnesota',
    'Mississippi',
    'Missouri',
    'Montana',
    'Nebraska',
    'Nevada',
    'New Hampshire',
    'New Jersey',
    'New Mexico',
    'New York',
    'North Carolina',
    'North Dakota',
    'Ohio',
    'Oklahoma',
    'Oregon',
    'Pennsylvania',
    'Rhode Island',
    'South Carolina',
    'South Dakota',
    'Tennessee',
    'Texas',
    'Utah',
    'Vermont',
    'Virginia',
    'Washington',
    'West Virginia',
    'Wisconsin',
    'Wyoming',
  ];

  constructor() {
    this.stateCtrl = new FormControl();
    this.stateCtrl2 = new FormControl();
    this.stateCtrl3 = new FormControl();
    this.filteredStates = this.stateCtrl.valueChanges
```

```
        .startWith(null)
        .map(name => this.filterStates(name));
    this.filteredStates = this.stateCtrl2.valueChanges
        .startWith(null)
        .map(name => this.filterStates(name));
    this.filteredStates = this.stateCtrl3.valueChanges
        .startWith(null)
        .map(name => this.filterStates(name));
  }

  filterStates(val: string) {
    return val ? this.states.filter(s => s.toLowerCase().indexOf(val.toLowerCase()) === 0)
               : this.states;
  }

}
```

Live Example

Read md-autocomplete online: https://riptutorial.com/angular-material2/topic/10850/md-autocomplete

Chapter 3: md-button

Introduction

This topic includes examples on how to create a button and what its' directives and other stuff do.

Parameters

Attribute	Description
`md-button`	Creates a rectangular button w/ no elevation.
`md-raised-button`	Creates a rectangular button w/ elevation
`md-icon-button`	Creates a circular button with a transparent background, meant to contain an icon
`md-fab`	Creates a circular button w/ elevation, defaults to theme's accent color
`md-mini-fab`	Same as `md-fab` but smaller
`disableRipple`	Whether the ripple effect for the button is disabled.

Remarks

For more information and more examples, visit the docs.

Examples

Simple buttons

To create a button, use the `md-button` attribute for a flat button and `md-raised-button` for an elevated button:

```
<button md-button>Button</button>
<button md-raised-button>Raised Button</button>
<button md-fab><md-icon>add</md-icon></button>
<button md-mini-fab><md-icon>check</md-icon></button>
<button md-icon-button><md-icon>person_add</md-icon></button>
```

Plunker Demo

For more information about icons, see the docs on `md-icon`.

Read md-button online: https://riptutorial.com/angular-material2/topic/10870/md-button

Chapter 4: md-datepicker

Introduction

This topic focuses on examples related to md-datepicker.

Remarks

For more details, please check the `md-datepicker` documentation here.

Examples

Data binding with md-datapicker

datepicker-overview-example.html:

```
<md-input-container>
  <input mdInput
         [mdDatepicker]="picker"
         [(ngModel)]="date"
         placeholder="Choose a date">
  <button mdSuffix [mdDatepickerToggle]="picker"></button>
</md-input-container>
<md-datepicker  #picker></md-datepicker>
<div>
  Date Chosen using 'ngModel':
  <div>{{ date }}</div>
</div>
```

datepicker-overview-example.ts:

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

@Component({
  selector: 'datepicker-overview-example',
  templateUrl: 'datepicker-overview-example.html'
})
export class DatepickerOverviewExample implements OnInit {

  date;

  ngOnInit(){
    this.date = new Date();
  }

}
```

Live demo

Passing selected date value to a function using $event

datepicker-overview-example.html:

```
<md-input-container>
    <input mdInput [mdDatepicker]="picker" placeholder="Choose a date" [(ngModel)]="value">
    <button mdSuffix [mdDatepickerToggle]="picker"></button>
</md-input-container>
<md-datepicker #picker [startAt]="startDate" (selectedChanged)="selectedDate($event)"></md-
datepicker>

<p>ngModel Value: {{value}}</p>

<p>Date from selectedDate(): {{checkDate}}</p>
```

datepicker-overview-example.ts:

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

@Component({
  selector: 'datepicker-overview-example',
  templateUrl: 'datepicker-overview-example.html'
})
export class DatepickerOverviewExample {

  value: Date = new Date();

  checkDate: Date;

  selectedDate(date){
   // ngModel still returns the old value
      console.log("ngModel: " + this.value);

   // date passes the newly selected value
      console.log("Selected Value: " + date);
      this.checkDate = date;
  }
}
```

Live demo

Open datepicker on focus

This example also includes the use of properties:

- min
- max
- startAt
- startView
- touchUi

datepicker-overview-example.html:

```
<h2>Options</h2>
<p>
  <md-checkbox [(ngModel)]="touch">Use touch UI</md-checkbox>
  <md-checkbox [(ngModel)]="filterOdd">Filter odd months and dates</md-checkbox>
```

```
  <md-checkbox [(ngModel)]="yearView">Start in year view</md-checkbox>
</p>
<p>
  <md-input-container>
    <input mdInput [mdDatepicker]="minDatePicker" [(ngModel)]="minDate" placeholder="Min date"
(keydown)="false" (click)="minDatePicker.open()">
    <button mdSuffix [mdDatepickerToggle]="minDatePicker"></button>
  </md-input-container>
  <md-datepicker #minDatePicker [touchUi]="touch"></md-datepicker>
  <md-input-container>
    <input mdInput [mdDatepicker]="maxDatePicker" [(ngModel)]="maxDate" placeholder="Max date"
(keydown)="false" (focus)="maxDatePicker.open()">
    <button mdSuffix [mdDatepickerToggle]="maxDatePicker"></button>
  </md-input-container>
  <md-datepicker #maxDatePicker [touchUi]="touch"></md-datepicker>
</p>
<p>
  <md-input-container>
    <input mdInput [mdDatepicker]="startAtPicker" [(ngModel)]="startAt" placeholder="Start at
date" (keydown)="false" (focus)="startAtPicker.open()">
    <button mdSuffix [mdDatepickerToggle]="startAtPicker"></button>
  </md-input-container>
  <md-datepicker #startAtPicker [touchUi]="touch"></md-datepicker>
</p>

<h2>Result</h2>

<p>
  <button [mdDatepickerToggle]="resultPicker"></button>
  <md-input-container>
    <input mdInput
           #resultPickerModel="ngModel"
           [mdDatepicker]="resultPicker"
           [(ngModel)]="date"
           [min]="minDate"
           [max]="maxDate"
           [mdDatepickerFilter]="filterOdd ? dateFilter : null"
           placeholder="Pick a date"
           (keydown)="false"
           (focus)="resultPicker.open()">
    <md-error *ngIf="resultPickerModel.hasError('mdDatepickerMin')">Too early!</md-error>
    <md-error *ngIf="resultPickerModel.hasError('mdDatepickerMax')">Too late!</md-error>
    <md-error *ngIf="resultPickerModel.hasError('mdDatepickerFilter')">Date unavailable!</md-
error>
  </md-input-container>
  <md-datepicker
      #resultPicker
      [touchUi]="touch"
      [startAt]="startAt"
      [startView]="yearView ? 'year' : 'month'">
  </md-datepicker>
</p>
<p>
  <input [mdDatepicker]="resultPicker2"
         [(ngModel)]="date"
         [min]="minDate"
         [max]="maxDate"
         [mdDatepickerFilter]="filterOdd ? dateFilter : null"
         (focus)="resultPicker2.open()"
         placeholder="Pick a date"
         (keydown)="false">
```

```
  <button [mdDatepickerToggle]="resultPicker2"></button>
  <md-datepicker
      #resultPicker2
      [touchUi]="touch"
      [startAt]="startAt"
      [startView]="yearView ? 'year' : 'month'">
  </md-datepicker>
</p>
```

datepicker-overview-example.ts:

```
import {Component, OnInit} from '@angular/core';
import {MdDatepicker} from '@angular/material';

@Component({
  selector: 'datepicker-overview-example',
  templateUrl: 'datepicker-overview-example.html'
})
export class DatepickerOverviewExample implements OnInit {

  touch: boolean;
  filterOdd: boolean;
  yearView: boolean;
  minDate: Date;
  maxDate: Date;
  startAt: Date;
  date: Date;
  dateFilter = (date: Date) => date.getMonth() % 2 == 1 && date.getDate() % 2 == 0;

}
```

Live demo

Set different locale for md-datepicker

This example requires importing `DateAdapter`.

```
import {DateAdapter} from '@angular/material';
```

datepicker.component.html:

```
<md-input-container>
  <input mdInput [mdDatepicker]="picker" placeholder="Choose a date">
  <button mdSuffix [mdDatepickerToggle]="picker"></button>
</md-input-container>
<md-datepicker #picker></md-datepicker>

<p></p>
<div>
  <button md-raised-button (click)="setLocale('en')">English - US</button>

  <button md-raised-button (click)="setLocale('es')">Spanish</button>

  <button md-raised-button (click)="setLocale('zh')">Chinese</button>

  <button md-raised-button (click)="setLocale('nl')">Dutch</button>
```

```
  <button md-raised-button (click)="setLocale('bn')">Bengali</button>

  <button md-raised-button (click)="setLocale('hi')">Hindi</button>

  <button md-raised-button (click)="setLocale('ar')">Arabic</button>
</div>
```

datepicker.component.ts:

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

@Component({
  selector: 'datepicker-overview-example',
  templateUrl: './datepicker-overview-example.html',
  styleUrls: ['./datepicker-overview-example.css'],
})
export class DatepickerOverviewExample {

  constructor(private dateAdapter: DateAdapter<Date>) {
    this.dateAdapter.setLocale('en');
  }

  setLocale(val){
    console.log(val);
    this.dateAdapter.setLocale(val);
  }

}
```

Live demo

A list of locale language code can be found here.

Read md-datepicker online: https://riptutorial.com/angular-material2/topic/10876/md-datepicker

Chapter 5: md-dialog

Introduction

This topic includes examples of `md-dialog`.

Remarks

To find more details on `md-dialog`, please check the documentation here.

Examples

Initialize md-dialog with data passed from parent component

This example requires `MdDialogRef` and `MD_DIALOG_DATA`. Please import them in the component module.

```
import {MdDialog, MdDialogRef, MD_DIALOG_DATA} from '@angular/material';
```

input-overview-example.html:

```
<md-input-container>
  <input mdInput
         [(ngModel)]="id"
         placeholder="Value passed to md-dialog">
</md-input-container>

<p></p>

<button md-raised-button
     (click)="openDialog(id)">
  Open Dialog
</button>
```

input-overview-example.ts:

```
import {Component, Inject, Input, OnInit } from '@angular/core';
import {MdDialog, MdDialogRef, MD_DIALOG_DATA} from '@angular/material';

@Component({
  selector: 'input-overview-example',
  templateUrl: 'input-overview-example.html'
})
export class InputOverviewExample {

  id: any;

  @Input() isChecked: boolean;

  constructor(public dialog: MdDialog) {}
```

```
    openDialog(value) {
      let dialogRef = this.dialog.open(DialogResultExampleDialog, {
        data: {
          id: value
        }
      });
      dialogRef.afterClosed().subscribe(result => {
        console.log(result);
      });
    }
}

@Component({
  selector: 'dialog-result-example-dialog',
  template: `<p md-dialog-title>Confirm Toggle </p>
             <p md-dialog-content>Id passed from component: {{ this.passedId }}</p>
             <md-dialog-actions>
                <button md-button color="primary"
(click)="dialogRef.close('Cancel')">Cancel</button>
                <button md-button color="primary"
(click)="dialogRef.close('continue')">Continue</button>
            </md-dialog-actions>
          `,
})
export class DialogResultExampleDialog implements OnInit {

  passedId: string;

  constructor(@Inject(MD_DIALOG_DATA) private data: { id: string },
              public dialogRef: MdDialogRef<DialogResultExampleDialog>) {}

  ngOnInit(){
    this.passedId = this.data.id;
  }
}
```

Live demo

Read md-dialog online: https://riptutorial.com/angular-material2/topic/10877/md-dialog

Chapter 6: md-icon

Examples

Creating an icon

The following is a guide on how to create an icon from material design icons:

1. Load the icon font from Google CDN in your `index.html`:

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

   Alternatively, you may import it in your `styles.css`:

   ```
   @import url('https://fonts.googleapis.com/icon?family=Material+Icons');
   ```

2. Use it as follows:

   ```
   <md-icon>menu</md-icon>
   ```

You're done!

Using SVG Icons

This example shows how to use SVG icons in your app.

1. Download the SVG iconset / icon (in this case, we're getting the icons from https://materialdesignicons.com/getting-started.
2. Save it under your `assets` folder or somewhere else which you can access with.
3. In your `app.module.ts`, add the following code:

   ```
   import { MdIconRegistry } from '@angular/material';
   import { DomSanitizer } from '@angular/platform-browser';

   export class AppModule {
       constructor(mdIconRegistry: MdIconRegistry, domSanitizer: DomSanitizer){
           // Note that you have to sanitize the resource since Angular will complain that
   it will cause XSS problems.
           // More info: https://g.co/ng/security#xss

   mdIconRegistry.addSvgIconSet(domSanitizer.bypassSecurityTrustResourceUrl('assets/icons.svg'))

       }
   }
   ```

4. Use it via the `svgIcon` attribute:

```
<md-icon svgIcon="menu"></md-icon>
```

Read md-icon online: https://riptutorial.com/angular-material2/topic/10868/md-icon

Chapter 7: md-table

Introduction

This topic includes examples related to md-table

Remarks

For more details on `md-table` please check the documentation

Examples

Connect DataSource from external API

Please me mindful of importing all necessary libraries required.

This example uses `InMemoryDbService` from `angular-in-memory-web-api` to provide the JSON data from mock API.

Live demo

service.ts:

```
import { Injectable }    from '@angular/core';
import { Headers, Http } from '@angular/http';
import 'rxjs/add/operator/toPromise';

@Injectable()
export class AppState {

  private headers = new Headers({'Content-Type': 'application/json'});
  private apiUrl = 'api/data';

  constructor(private http: Http) { }

  fetchFilterFields() {
    console.log(this.apiUrl);
    return this.http.get(this.apiUrl)
               .toPromise()
               .then(response => response.json().data)
               .catch(this.handleError);
  }

  private handleError(error: any): Promise<any> {
    console.error('An error occurred', error); // for demo purposes only
    return Promise.reject(error.message || error);
  }

}
```

component.ts:

```
import {Component} from '@angular/core';
import {DataSource} from '@angular/cdk';
import {BehaviorSubject} from 'rxjs/BehaviorSubject';
import {Observable} from 'rxjs/Observable';
import 'rxjs/add/observable/merge';

import { AppState } from './shared.service';

@Component({
  selector: 'md-table-example',
  templateUrl: 'select-form-example.html',
  styleUrls: ['select-form-example.css']
})
export class SelectFormExample implements OnInit {

  displayedColumns = ['id', 'name'];
  dataSource: ExampleDataSource | null;

  constructor(private appState: AppState){ }

  ngOnInit(){
    this.dataSource = new ExampleDataSource(this.appState);
  }

}

export interface UserData {
  id: string;
  name: string;
}

export class ExampleDataSource extends DataSource<any> {
  constructor(private appState: AppState) {
    super();
  }

  subject: BehaviorSubject<Hero[]> = new BehaviorSubject<Hero[]>([]);

  connect(): Observable<Hero[]> {
      console.log('connect');
      if (!this.subject.isStopped)
          this.appState.fetchFilterFields()
              .then(res => {
                  this.subject.next(res)
              });
      return Observable.merge(this.subject);
  }

  disconnect() {
      this.subject.complete();
      this.subject.observers = [];
  }
}
```

component.html:

```
<h2> Material Table </h2>

<div *ngIf="dataSource" class="example-container mat-elevation-z8">
  <md-table #table [dataSource]="dataSource">
```

```
    <!-- ID Column -->
    <ng-container cdkColumnDef="id">
      <md-header-cell *cdkHeaderCellDef> ID </md-header-cell>
      <md-cell *cdkCellDef="let row"> {{row.id}} </md-cell>
    </ng-container>

    <!-- Name Column -->
    <ng-container cdkColumnDef="name">
      <md-header-cell *cdkHeaderCellDef> Name </md-header-cell>
      <md-cell *cdkCellDef="let row"> {{row.name}} </md-cell>
    </ng-container>

    <md-header-row *cdkHeaderRowDef="displayedColumns"></md-header-row>
    <md-row *cdkRowDef="let row; columns: displayedColumns;"></md-row>
  </md-table>
</div>
```

For reference:

in-memory-data-service.ts:

```
import { InMemoryDbService } from 'angular-in-memory-web-api';
export class InMemoryDataService implements InMemoryDbService {
  createDb() {
    const data = [
      { id: 1,  name: 'Ironcast' },
      { id: 2, name: 'Mr. Nice' },
      { id: 3, name: 'Narco' },
      { id: 4, name: 'Bombasto' },
      { id: 5, name: 'Celeritas' },
      { id: 6, name: 'Magneta' },
      { id: 7, name: 'RubberMan' },
      { id: 8, name: 'Dynama' },
      { id: 9, name: 'Dr. IQ' },
      { id: 10, name: 'Magma' },
      { id: 11, name: 'Tornado' }
    ];
    return {data};
  }
}
```

Read md-table online: https://riptutorial.com/angular-material2/topic/10911/md-table

Credits

S. No	Chapters	Contributors
1	Getting started with angular-material2	Community, Edric, Maciej Treder, Nehal
2	md-autocomplete	Nehal
3	md-button	Edric
4	md-datepicker	Nehal
5	md-dialog	Nehal
6	md-icon	Edric
7	md-table	Nehal