Angular Material UI

What is angular material UI?

Angular Material is a UI component library for Angular applications. It provides a set of pre-built, customizable UI components following the Material Design guidelines developed by Google. Material Design is a design language that emphasizes clean, modern, and consistent design principles across different platforms and devices.

Angular Material offers a wide range of components such as buttons, cards, forms, dialogs, menus, and more, which can be easily integrated into Angular applications to create responsive and visually appealing user interfaces. These components are built with Angular directives and utilize Angular's powerful features like data binding, dependency injection, and animations.

Using Angular Material helps developers to save time and effort in building UI components from scratch, as well as ensuring consistency and adherence to design best practices.

What are angular material components?

Angular Material offers a wide range of UI components that developers can utilize to build modern and responsive web applications. Some of the key Angular Material components include:

1. **Buttons:** Various types of buttons such as flat, raised, icon, fab (floating action button), toggle, and more.
2. **Cards:** Containers for organizing content into a consistent format, often used to display information or media.
3. **Forms:** Components for building forms, including inputs, selects, checkboxes, radio buttons, sliders, and date pickers.
4. **Dialogs:** Modal dialogs for displaying important messages, notifications, or prompting user interactions.
5. **Menus:** Dropdown menus and context menus for navigation or providing additional options.
6. **Toolbars:** Flexible toolbar components for creating headers, footers, or navigation bars.
7. **Lists:** Lists for displaying collections of items, with support for various layouts and interactive features.
8. **Grid List:** A layout component for creating responsive grids of items, commonly used for displaying images or tiles.
9. **Tabs:** Tabbed navigation for organizing content into separate sections or views.
10. **Expansion Panels:** Components for creating collapsible panels that reveal additional content when clicked or tapped.

11. **Steppers:** Step-by-step wizards or progress trackers for guiding users through a sequence of tasks.
12. **Snackbar:** Lightweight notifications that appear at the bottom of the screen, often used for displaying temporary messages or alerts.
13. **Progress Spinners:** Animated indicators of loading or processing, available in various styles and sizes.
14. **Chips:** Compact elements for representing complex information, such as tags or contact details.
15. **Tooltip:** Components for adding tooltips to elements, providing additional context or explanation.
16. **Data Tables:** Components for displaying data in tabular format with sorting, filtering, and pagination capabilities.

These components help developers create responsive, accessible, and visually appealing user interfaces with minimal effort, as they come with built-in styles and functionality that can be easily customized and extended to suit application needs.

Example 1: Create different types of buttons in angular material?

Step1:

First, make sure you have Angular Material installed in your project.

If not, you can install it using Angular CLI command

ng add @angular/material

Step2:

Make sure you import MatButtonModule, MatIconModule in your app.module.ts file.

Here we are using Icons also that's why MatIconModule imported.

These will enable Angular Material buttons and icons in your Angular application.

```
import { NgModule } from '@angular/core';
import { BrowserModule, provideClientHydration } from '@angular/platform-browser';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { MatButtonModule } from '@angular/material/button';
import { MatIconModule } from '@angular/material/icon';
```

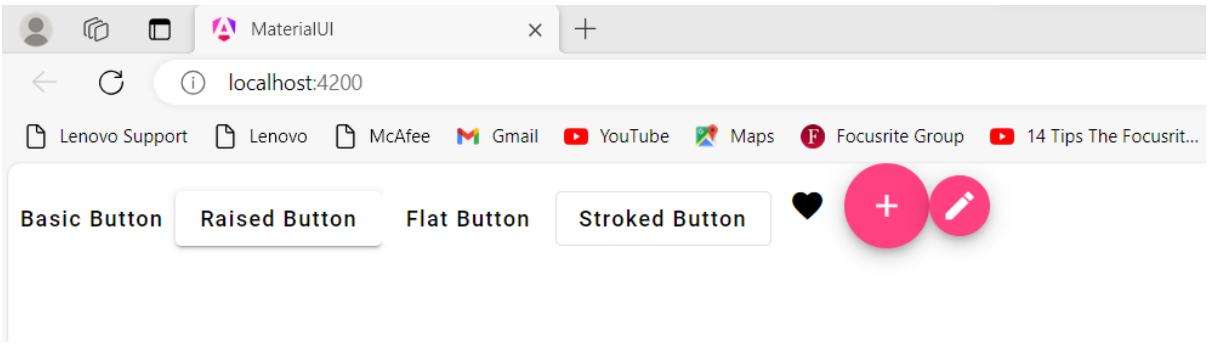
```
@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    BrowserAnimationsModule,
    MatButtonModule,
    MatIconModule
  ],
  providers: [
    provideClientHydration()
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Step3:

Then, you can use Angular Material buttons in your component template like this:

```
<!-- app.component.html -->
<button mat-button>Basic Button</button>
<button mat-raised-button>Raised Button</button>
<button mat-flat-button>Flat Button</button>
<button mat-stroked-button>Stroked Button</button>
<button mat-icon-button><mat-icon>favorite</mat-icon></button>
<button mat-fab><mat-icon>add</mat-icon></button>
<button mat-mini-fab><mat-icon>edit</mat-icon></button>
```

Output:



Explanation:

In the above example we used different types of buttons

mat-button: Basic button style.

mat-raised-button: Button with a raised effect.

mat-flat-button: Button with a flat style.

mat-stroked-button: Button with a stroked style.

mat-icon-button: Button with an icon.

mat-fab: Floating action button.

mat-mini-fab: Mini floating action button.

You can customize these buttons further using Angular Material's theming and styling features.

Exampe 2: create different types of buttons with different styles.

Step1:

First, make sure you have Angular Material installed in your project.

If not, you can install it using Angular CLI command

ng add @angular/material

Step2:

Make sure you import MatButtonModule, MatIconModule, MatDividerModule in your app.module.ts file.

Here we are using Icons and divider also that's why MatIconModule, MatDividerModule imported.

These will enable Angular Material buttons and icons in your Angular application.

```
import { NgModule } from '@angular/core';
import { BrowserModule, provideClientHydration } from '@angular/platform-browser';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { MatButtonModule } from '@angular/material/button';
import { MatIconModule } from '@angular/material/icon';
import { MatDividerModule } from '@angular/material/divider';
@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    BrowserAnimationsModule,
    MatButtonModule,
    MatIconModule,
    MatDividerModule
  ],
  providers: [
    provideClientHydration()
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Step3:

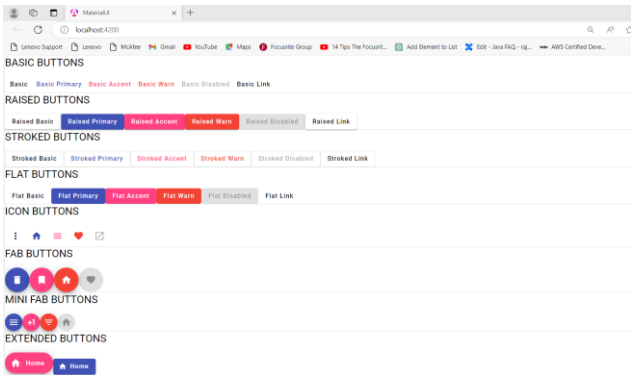
Then, you can use Angular Material buttons in your component template like this:

```
<h1>BASIC BUTTONS</h1>
    <button mat-button>Basic</button>
    <button mat-button color="primary">Basic Primary</button>
    <button mat-button color="accent">Basic Accent</button>
    <button mat-button color="warn">Basic Warn</button>
    <button mat-button disabled>Basic Disabled</button>
    <a mat-button href="https://www.google.com/" target="_blank">Basic Link</a>
<mat-divider></mat-divider>
<h1>RAISED BUTTONS</h1>
    <button mat-raised-button>Raised Basic</button>
    <button mat-raised-button color="primary">Raised Primary</button>
    <button mat-raised-button color="accent">Raised Accent</button>
    <button mat-raised-button color="warn">Raised Warn</button>
    <button mat-raised-button disabled>Raised Disabled</button>
    <a mat-raised-button href="https://www.google.com/" target="_blank">Raised
Link</a>
<mat-divider></mat-divider>
<h1>STROKED BUTTONS</h1>
    <button mat-stroked-button>Stroked Basic</button>
    <button mat-stroked-button color="primary">Stroked Primary</button>
    <button mat-stroked-button color="accent">Stroked Accent</button>
    <button mat-stroked-button color="warn">Stroked Warn</button>
    <button mat-stroked-button disabled>Stroked Disabled</button>
    <a mat-stroked-button href="https://www.google.com/" target="_blank">Stroked
Link</a>
<mat-divider></mat-divider>
<h1>FLAT BUTTONS</h1>
    <button mat-flat-button>Flat Basic</button>
    <button mat-flat-button color="primary">Flat Primary</button>
```

```
    <button mat-flat-button color="accent">Flat Accent</button>
    <button mat-flat-button color="warn">Flat Warn</button>
    <button mat-flat-button disabled>Flat Disabled</button>
    <a mat-flat-button href="https://www.google.com/" target="_blank">Flat
Link</a>
<mat-divider></mat-divider>
<h1>ICON BUTTONS</h1>
    <button mat-icon-button>
      <mat-icon>more_vert</mat-icon>
    </button>
    <button mat-icon-button color="primary">
      <mat-icon>home</mat-icon>
    </button>
    <button mat-icon-button color="accent">
      <mat-icon>menu</mat-icon>
    </button>
    <button mat-icon-button color="warn">
      <mat-icon>favorite</mat-icon>
    </button>
    <button mat-icon-button disabled>
      <mat-icon>open_in_new</mat-icon>
    </button>
<mat-divider></mat-divider>
<h1>FAB BUTTONS</h1>
      <button mat-fab color="primary">
        <mat-icon>delete</mat-icon>
      </button>
      <button mat-fab color="accent">
        <mat-icon>bookmark</mat-icon>
      </button>
```

```
    <button mat-fab color="warn">
      <mat-icon>home</mat-icon>
    </button>
    <button mat-fab disabled>
      <mat-icon>favorite</mat-icon>
    </button>
<mat-divider></mat-divider>
<h1>MINI FAB BUTTONS</h1>
    <button mat-mini-fab color="primary">
      <mat-icon>menu</mat-icon>
    </button>
    <button mat-mini-fab color="accent">
      <mat-icon>plus_one</mat-icon>
    </button>
    <button mat-mini-fab color="warn">
      <mat-icon>filter_list</mat-icon>
    </button>
    <button mat-mini-fab disabled>
      <mat-icon>home</mat-icon>
    </button>
<mat-divider></mat-divider>
<h1>EXTENDED BUTTONS</h1>
    <button mat-fab extended>
      <mat-icon>home</mat-icon>
     Home
    </button>
    <button mat-raised-button color="primary">
      <mat-icon>home</mat-icon>
     Home
    </button>
```

Output:



Explanation:

Angular Material provides a variety of pre-defined colors for buttons, which can be customized to fit your application's design scheme. These colors are part of Angular Material's theming system and can be easily applied to buttons using Angular Material's components.

Here are some of the commonly used color palettes provided by Angular Material:

Primary: This is typically used for primary actions or buttons that denote the primary purpose of an interaction.

Accent: Accent color is used for elements that need to stand out and grab user attention. It's often used for secondary actions or elements.

Warn: Warn color is typically used for indicating errors or alerts. It's commonly associated with actions that may have negative consequences.

You can use these colors by applying Angular Material's color directives to your buttons.

Traditional fab buttons are circular and only have space for a single icon. However, you can add the extended attribute to allow the fab to expand into a rounded rectangle shape with space for a text label in addition to the icon. Only full sized fabs support the extended attribute, mini fabs do not.

Example 3: create single and multiselecting toggle buttons.

Step 1:

First, make sure you have Angular Material installed in your project.

If not, you can install it using Angular CLI command

ng add @angular/material

Step 2:

Make sure you import MatButtonToggleModule, MatButtonToggleGroup in your app.module.ts file.

These will enable Angular Material toggle buttons and material toggle group in your Angular application.

```
import { NgModule } from '@angular/core';
import { BrowserModule, provideClientHydration } from '@angular/platform-browser';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { MatButtonToggleModule, MatButtonToggleGroup } from '@angular/material/button-toggle'

@NgModule({
 declarations: [
  AppComponent,
 ],
 imports: [
  BrowserModule,
  AppRoutingModule,
  BrowserAnimationsModule,
  MatButtonToggleModule,
  MatButtonToggleGroup
 ],
```

```
  providers: [
    provideClientHydration()
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

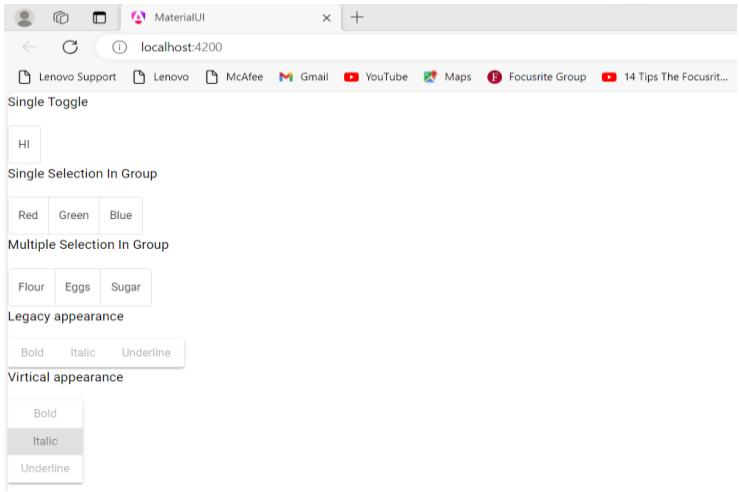
Step 3:

You can use Angular Material toggle buttons in your component template like this:

```
<h3>Single Toggle</h3>
    <mat-button-toggle value="hi">HI</mat-button-toggle>
<h3>Single Selection In Group</h3>
    <mat-button-toggle-group name="favoriteColor">
      <mat-button-toggle value="red">Red</mat-button-toggle>
      <mat-button-toggle value="green">Green</mat-button-toggle>
      <mat-button-toggle value="blue">Blue</mat-button-toggle>
    </mat-button-toggle-group>

<h3>Multiple Selection In Group</h3>
    <mat-button-toggle-group name="ingredients" multiple>
      <mat-button-toggle value="flour">Flour</mat-button-toggle>
      <mat-button-toggle value="eggs">Eggs</mat-button-toggle>
      <mat-button-toggle value="sugar">Sugar</mat-button-toggle>
    </mat-button-toggle-group>

<h3>Legacy appearance</h3>
    <mat-button-toggle-group appearance="legacy" name="fontStyle">
      <mat-button-toggle value="bold">Bold</mat-button-toggle>
      <mat-button-toggle value="italic">Italic</mat-button-toggle>
      <mat-button-toggle value="underline">Underline</mat-button-toggle>
    </mat-button-toggle-group>
```

```
<h3>Virtical appearance</h3>
<mat-button-toggle-group appearance="legacy" name="fontStyle"
vertical="true">
<mat-button-toggle value="bold">Bold</mat-button-toggle>
<mat-button-toggle value="italic">Italic</mat-button-toggle>
<mat-button-toggle value="underline">Underline</mat-button-toggle>
</mat-button-toggle-group>
```

Output:



Explanation:

In Angular Material, MatButtonToggle and MatButtonToggleGroup are components used to create toggleable buttons within a group. They are particularly useful when you want users to select one option from a set of mutually exclusive options.

MatButtonToggle

MatButtonToggle represents a single toggle button within a group. It behaves like a regular button but maintains a selected state when clicked. Only one button within the same toggle group can be selected at a time.

value: The value attribute associated with the button. This is used to identify the selected option within the group.

disabled: Whether the button is disabled or not.

MatButtonToggleGroup

MatButtonToggleGroup is a container for MatButtonToggle components. It manages the selection state of the toggle buttons within it and ensures that only one button is selected at a time.

By default, mat-button-toggle-group acts like a radio-button group- only one item can be selected. In this mode, the value of the mat-button-toggle-group will reflect the value of the selected button and ngModel is supported.

Adding the multiple attribute allows multiple items to be selected (checkbox behavior). In this mode the values of the toggles are not used, the mat-button-toggle-group does not have a value, and ngModel is not supported.

By default, the appearance of mat-button-toggle-group and mat-button-toggle will follow the latest Material Design guidelines. If you want to, you can switch back to the appearance that was following the previous Material Design spec by using the appearance input.

The button-toggles can be rendered in a vertical orientation by adding the vertical attribute.

Example 4: Create different badges in angular material

Step 1:

First, make sure you have Angular Material installed in your project.

If not, you can install it using Angular CLI command

ng add @angular/material

Step 2:

Make sure you import MatBadgeModule, MatButtonModule, MatIconModule in your app.module.ts file.

These will enable Angular material badges, material buttons and material icons in your Angular application.

import { NgModule } from '@angular/core';

import { BrowserModule, provideClientHydration } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';

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

```
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { MatButtonModule } from '@angular/material/button';
import { MatIconModule } from '@angular/material/icon';
import { MatBadgeModule } from '@angular/material/badge'
@NgModule({
  declarations: [
    AppComponent,
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    BrowserAnimationsModule,
    MatButtonModule,
    MatIconModule,
    MatBadgeModule
  ],
  providers: [
    provideClientHydration()
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Step 3:

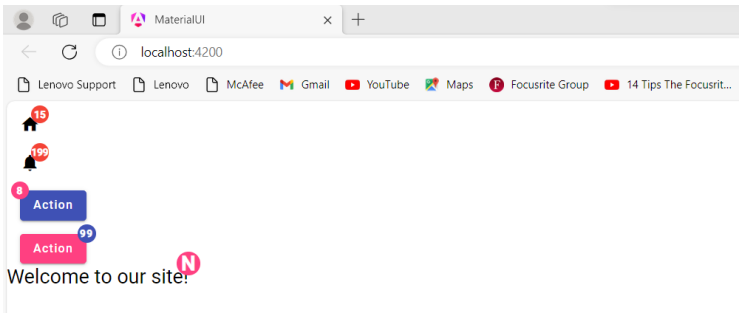
Create style for margin for components in css file.

```
.my-margin{
    margin-top: 16px;
    margin-left: 16px;
  }
```

You can use Angular Material badges along with buttons, icons and headings in your component template like this:

```
<div class="my-margin">
   <mat-icon matBadge="15" matBadgeColor="warn">home</mat-icon>
</div>

<div class="my-margin">
   <mat-icon matBadge="199" matBadgeColor="warn">notifications</mat-icon>
</div>



<div class="my-margin">
   <button mat-raised-button color="primary" matBadge="8"
matBadgePosition="before" matBadgeColor="accent">
      Action
   </button>
</div>

<div class="my-margin">
   <button mat-raised-button color="accent" matBadge="99"
matBadgeColor="primary">
      Action
   </button>
</div>

<div style="display: flex;">
   <h1 matBadge="N" matBadgeSize="large" matBadgePosition="above after"
matBadgeColor="accent">Welcome to our site!</h1>
</div>
```

Output:



Explanation:

Badges are small status descriptors for UI elements. A badge consists of a small circle, typically containing a number or other short set of characters, that appears in proximity to another object.

Badges must always be applied to block-level elements. Content that participates in block layout is called block-level content. Example headings, div, paragraphs, buttons etc.

Badge position:

By default, the badge will be placed above after. The direction can be changed by defining the attribute matBadgePosition follow by above|below and before|after.

The overlap of the badge in relation to its inner contents can also be defined using the matBadgeOverlap tag. Typically, you want the badge to overlap an icon and not a text phrase. By default it will overlap.

Badge sizing

The badge has 3 sizes: small, medium and large. By default, the badge is set to medium. You can change the size by adding matBadgeSize to the host element.

Badge Theming

Badges can be colored in terms of the current theme using the matBadgeColor property to set the background color to primary, accent, or warn.

Example 5: Create any two angular material cards and display them.

Step 1:
First, make sure you have Angular Material installed in your project.

If not, you can install it using Angular CLI command

ng add @angular/material

Step 2:
Make sure you import MatCardModule, MatButtonModule, MatIconModule in your app.module.ts file.

These will enable Angular material cards, material buttons and material icons in your Angular application.

```
import { NgModule } from '@angular/core';
import { BrowserModule, provideClientHydration } from '@angular/platform-browser';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { MatButtonModule } from '@angular/material/button';
import { MatIconModule } from '@angular/material/icon';
import { MatCardModule } from '@angular/material/card';
@NgModule({
 declarations: [
  AppComponent,
 ],
 imports: [
  BrowserModule,
  AppRoutingModule,
  BrowserAnimationsModule,
  MatButtonModule,
  MatIconModule,
  MatCardModule
```

```
  ],
  providers: [
    provideClientHydration()
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Step 3:

Create styles for cards and images in css file.

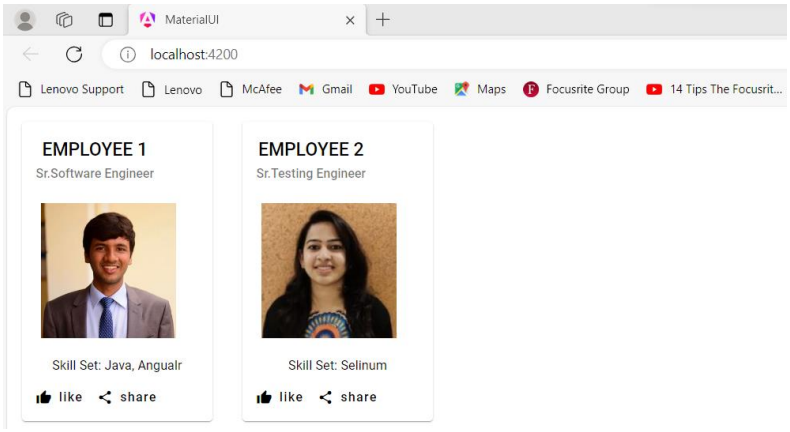
```
.mycard-style{
    margin : 1%;
    width : 13%;
    text-align : center ;
    float : left;
}
.myimage-style{
    margin : 10%;
    float : left;
}
```

You can use Angular Material cards along with buttons, icons and images in your component template like this:

```
<mat-card class="mycard-style">
  <mat-card-header>
    <mat-card-title>EMPLOYEE 1</mat-card-title>
    <mat-card-subtitle>Sr.Software Engineer</mat-card-subtitle>
  </mat-card-header>
  <img src="assets/emp1.jpg" mat-card-lg-image class="myimage-style" />
  <mat-card-content>Skill Set: Java, Angualr</mat-card-content>
  <mat-card-actions>
    <button mat-button><mat-icon>thumb_up</mat-icon>like</button>
```

```
        <button mat-button><mat-icon>share</mat-icon>share</button>
    </mat-card-actions>
    <mat-card-footer></mat-card-footer>
</mat-card>

<mat-card class="mycard-style">
    <mat-card-header>
        <mat-card-title>EMPLOYEE 2</mat-card-title>
        <mat-card-subtitle>Sr.Testing Engineer</mat-card-subtitle>
    </mat-card-header>
    <img src="assets/emp2.jpg" mat-card-lg-image class="myimage-style"/>
    <mat-card-content>Skill Set: Selinum</mat-card-content>
    <mat-card-actions>
        <button mat-button><mat-icon>thumb_up</mat-icon>like</button>
        <button mat-button><mat-icon>share</mat-icon>share</button>
    </mat-card-actions>
    <mat-card-footer></mat-card-footer>
</mat-card>
```

Output:



Explanation:

MatCardModule specifically provides the MatCard component, which is a container component designed to display content in a card-like format.

The most basic card needs only an <mat-card> element with some content.

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

Angular Material provides a number of preset sections that you can use inside a <mat-card> as follows.

<mat-card-header>	Section anchored to the top of the card (adds padding)
<mat-card-content>	Primary card content (adds padding)
<img mat-card-image>	Card image. Stretches the image to the container width
<mat-card-actions>	Container for buttons at the card bottom (adds padding)
<mat-card-footer>	Section anchored to the bottom of the card

The <mat-card> element itself does not add any padding around its content. This allows developers to customize the padding to their liking by applying padding to the elements they put in the card.

Card headers:

A <mat-card-header> can contain any content, but there are several predefined elements that can be used to create a rich header to a card. These include follows.

<mat-card-title>	A title within the header
<mat-card-subtitle>	A subtitle within the header
<img mat-card-avatar>	An image used as an avatar within the header

In addition to using <mat-card-title> and <mat-card-subtitle> directly within the <mat-card-header>, they can be further nested inside a <mat-card-title-group> in order arrange them with a (non-avatar) image.

<mat-card-title-group> can be used to combine a title, subtitle, and image into a single section. This element can contain:

<mat-card-title>
<mat-card-subtitle>
One of:

<img mat-card-sm-image>
<img mat-card-md-image>
<img mat-card-lg-image>

Example 6: Create different lists in angular material

Step 1:
First, make sure you have Angular Material installed in your project.

If not, you can install it using Angular CLI command

ng add @angular/material

Step 2:
Make sure you import MatListModule, MatIconModule in your app.module.ts file.

These will enable Angular material lists and material icons in your Angular application.

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

import { BrowserModule, provideClientHydration } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';

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

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

import { MatIconModule } from '@angular/material/icon';

import { MatListModule } from '@angular/material/list';


@NgModule({
  declarations: [
    AppComponent,
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    BrowserAnimationsModule,
    MatIconModule,
    MatListModule
  ],
```

```
  providers: [
    provideClientHydration()
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

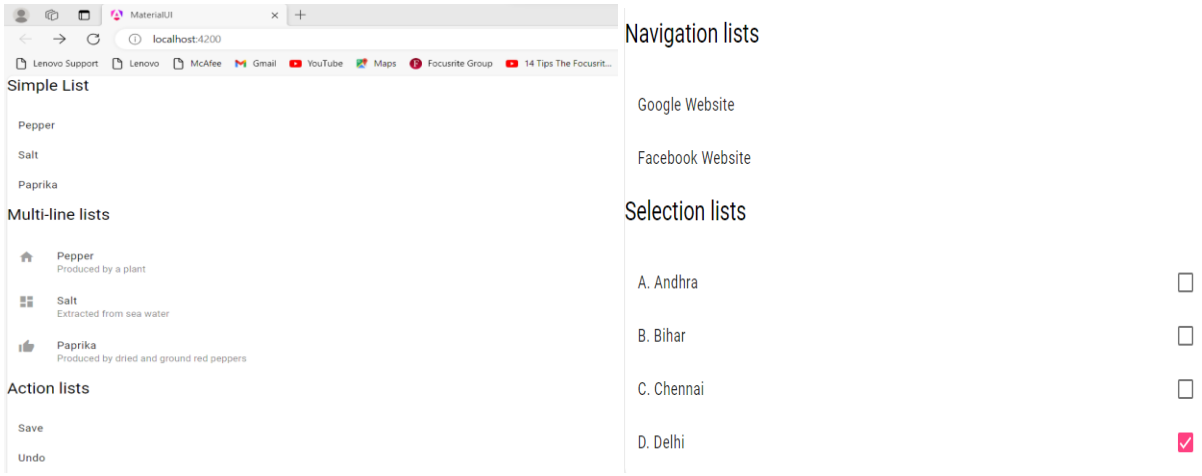
Step 3:

You can use Angular Material lists along with lists and icons in your component template like this:

```
<h1>Simple List</h1>
<mat-list>
    <mat-list-item>Pepper</mat-list-item>
    <mat-list-item>Salt</mat-list-item>
    <mat-list-item>Paprika</mat-list-item>
</mat-list>
<h1>Multi-line lists</h1>
  <mat-list>
    <mat-list-item>
      <span matListItemIcon><mat-icon>home</mat-icon></span>
      <span matListItemTitle>Pepper</span>
      <span matListItemLine>Produced by a plant</span>
    </mat-list-item>
    <mat-list-item>
      <span matListItemIcon><mat-icon>dashboard</mat-icon></span>
      <span matListItemTitle>Salt</span>
      <span matListItemLine>Extracted from sea water</span>
    </mat-list-item>
    <mat-list-item>
      <span matListItemIcon><mat-icon>thumb_up</mat-icon></span>
      <span matListItemTitle>Paprika</span>
```

```
    <span matListItemLine>Produced by dried and ground red peppers</span>
  </mat-list-item>
</mat-list>
<h1>Action lists</h1>
<mat-action-list>
  <button mat-list-item>Save</button>
  <button mat-list-item>Undo</button>
</mat-action-list>
<h1>Navigation lists</h1>
<mat-nav-list>
    <a mat-list-item href="https://www.google.com/">Google Website</a>
    <a mat-list-item href="https://www.facebook.com/">Facebook Website</a>
</mat-nav-list>
<h1>Selection lists</h1>
<mat-selection-list #shoes>
    <mat-list-option>A. Andhra</mat-list-option>
    <mat-list-option>B. Bihar</mat-list-option>
    <mat-list-option>C. Chennai</mat-list-option>
    <mat-list-option>D. Delhi</mat-list-option>
</mat-selection-list>
```

Output:



Explanation:

The MatListModule specifically provides components and directives for creating lists in Angular applications.

<mat-list> is a container component that wraps a series of <mat-list-item>.

Simple lists:

If a list item needs to show a single line of textual information, the text can be inserted directly into the <mat-list-item> element.

Multi-line lists:

List items that have more than one line of text have to use the matListItemTitle directive to indicate their title text for accessibility purposes, in addition to the matListItemLine directive for each subsequent line of text. matListItemIcon directive to add icon to list item.

The following directives can be used to style the content of a list item:

matListItemTitle →Indicates the title of the list item. Required for multi-line list items.

matListItemLine →Wraps a line of text within a list item.

matListItemIcon →Icon typically placed at the beginning of a list item.

matListItemAvatar→Image typically placed at the beginning of a list item.

matListItemMeta →Inserts content in the meta section at the end of a list item.

Action lists:

Use the <mat-action-list> element when each item in the list performs some action. Each item in an action list is a <button> element.

Simple action lists can use the mat-list-item attribute on button tag elements directly.

Navigation lists:

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

Simple navigation lists can use the mat-list-item attribute on anchor tag elements directly.

Selection lists:

A selection list provides an interface for selecting values, where each list item is an option.

The options within a selection-list should not contain further interactive controls, such as buttons and anchors.

Example 7: Create basic grid list in angular material

Step 1:
First, make sure you have Angular Material installed in your project.

If not, you can install it using Angular CLI command

ng add @angular/material

Step 2:
Make sure you import MatGridListModule in your app.module.ts file.

These will enable Angular material grid list in your Angular application.

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

import { BrowserModule, provideClientHydration } from '@angular/platform-
browser';

import { AppRoutingModule } from './app-routing.module';

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

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

import { MatGridListModule } from '@angular/material/grid-list';

@NgModule({

  declarations: [
    AppComponent,
  ],

  imports: [

    BrowserModule,

    AppRoutingModule,

    BrowserAnimationsModule,

    MatGridListModule

  ],

  providers: [

    provideClientHydration()
  ],

  bootstrap: [AppComponent]
})

export class AppModule { }
```

Step 3:

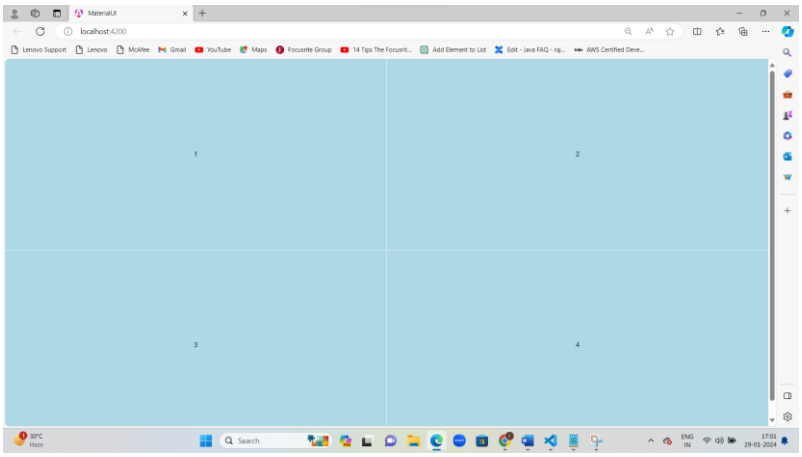
Create styles for grid background in css file.

```
mat-grid-tile {
    background: lightblue;
}
```

You can use Angular Material gridlist in your component template like this:

```
<mat-grid-list cols="2" rowHeight="2:1">
    <mat-grid-tile>1</mat-grid-tile>
    <mat-grid-tile>2</mat-grid-tile>
    <mat-grid-tile>3</mat-grid-tile>
    <mat-grid-tile>4</mat-grid-tile>
</mat-grid-list>
```

Output:



Explanation:

MatGridList component, which allows you to arrange content in a grid format with rows and columns.

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

Setting the number of columns:

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 of items.

Setting the row height:

The height of the rows in a grid list can be set via the rowHeight attribute. Row height for the list can be calculated in three ways:

Fixed height: The height can be in px, em, or rem. If no units are specified, px units are assumed (e.g. 100px, 5em, 250).

Ratio: This ratio is column-width:row-height, and must be passed in with a colon, not a decimal (e.g. 4:3).

Fit: Setting rowHeight to fit This mode automatically divides the available height by the number of rows. Please note the height of the grid-list or its container must be set.

If rowHeight is not specified, it defaults to a 1:1 ratio of width:height.

Setting the gutter size:

The gutter size can be set to any px, em, or rem value with the gutterSize property. If no units are specified, px units are assumed. By default the gutter size is 1px.

Example 8: Create grid list with different rowspan and colspan in angular material.

Step 1:
First, make sure you have Angular Material installed in your project.

If not, you can install it using Angular CLI command

ng add @angular/material

Step 2:
Make sure you import MatGridListModule in your app.module.ts file.

These will enable Angular material grid list in your Angular application.

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

import { BrowserModule, provideClientHydration } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';

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

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

import { MatGridListModule } from '@angular/material/grid-list';

@NgModule({
```

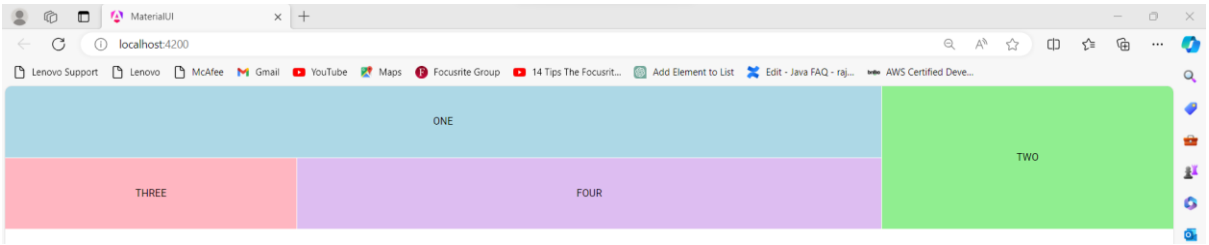
```
  declarations: [
    AppComponent,
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    BrowserAnimationsModule,
    MatGridListModule
  ],
  providers: [
    provideClientHydration()
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Step 3:

You can use Angular Material gridlist with colspan and rowspan in your component template like this:

```
<mat-grid-list cols="4" rowHeight="100px">
    <mat-grid-tile colspan="3" rowspan="1" style="background:lightblue">ONE</mat-grid-tile>
    <mat-grid-tile colspan="1" rowspan="2" style="background:lightgreen">TWO</mat-grid-tile>
    <mat-grid-tile colspan="1" rowspan="1" style="background:lightpink">THREE</mat-grid-tile>
    <mat-grid-tile colspan="2" rowspan="1" style="background:#DDBDF1">FOUR</mat-grid-tile>
  </mat-grid-list>
```

Output:



Explanation:

MatGridList component, which allows you to arrange content in a grid format with rows and columns.

Adding tiles that span multiple rows or columns:

It is possible to set the rowspan and colspan of each mat-grid-tile individually, using the rowspan and colspan properties. If not set, they both default to 1. The colspan must not exceed the number of cols in the mat-grid-list. There is no such restriction on the rowspan however, more rows will simply be added for it the tile to fill.

Tile headers and footers:

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

Example 9: Create grid list with different images in angular material.

Step 1:
First, make sure you have Angular Material installed in your project.

If not, you can install it using Angular CLI command

ng add @angular/material

Step 2:
Make sure you import MatGridListModule in your app.module.ts file.

These will enable Angular material grid list in your Angular application.

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

import { BrowserModule, provideClientHydration } from '@angular/platform-
browser';

import { AppRoutingModule } from './app-routing.module';

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

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

import { MatGridListModule } from '@angular/material/grid-list';

@NgModule({

  declarations: [
    AppComponent,
  ],

  imports: [
```

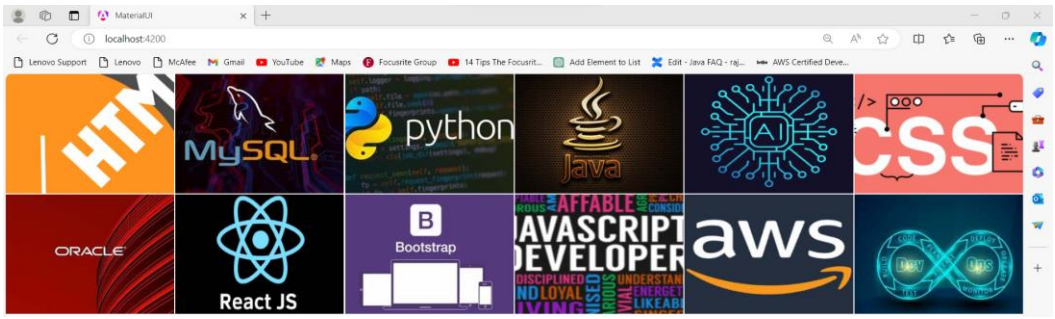
```
    BrowserModule,
    AppRoutingModule,
    BrowserAnimationsModule,
    MatGridListModule
  ],
  providers: [
    provideClientHydration()
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Step 3:

You can use Angular Material gridlist with different images in your component template like this:

```
<mat-grid-list cols="6" rowHeight="200px" gutterSize="2px">
    <mat-grid-tile><img src="assets/pics1.jpg" alt="Image"></mat-grid-tile>
    <mat-grid-tile><img src="assets/pics2.jpg" alt="Image"></mat-grid-tile>
    <mat-grid-tile><img src="assets/pics3.jpg" alt="Image"></mat-grid-tile>
    <mat-grid-tile><img src="assets/pics4.jpg" alt="Image"></mat-grid-tile>
    <mat-grid-tile><img src="assets/pics5.jpg" alt="Image"></mat-grid-tile>
    <mat-grid-tile><img src="assets/pics6.jpg" alt="Image"></mat-grid-tile>
    <mat-grid-tile><img src="assets/pics7.jpg" alt="Image"></mat-grid-tile>
    <mat-grid-tile><img src="assets/pics8.jpg" alt="Image"></mat-grid-tile>
    <mat-grid-tile><img src="assets/pics9.jpg" alt="Image"></mat-grid-tile>
    <mat-grid-tile><img src="assets/pics10.jpg" alt="Image"></mat-grid-tile>
    <mat-grid-tile><img src="assets/pics11.jpg" alt="Image"></mat-grid-tile>
    <mat-grid-tile><img src="assets/pics12.jpg" alt="Image"></mat-grid-tile>
</mat-grid-list>
```

Output:



Explanation:

MatGridList component, which allows you to arrange content in a grid format with rows and columns.

Adding tiles that span multiple rows or columns:

The gutter size can be set to any px, em, or rem value with the gutterSize property. If no units are specified, px units are assumed. By default the gutter size is 1px.

Example 10: Create grid list with image and card in angular material.

Step 1:
First, make sure you have Angular Material installed in your project.

If not, you can install it using Angular CLI command

ng add @angular/material

Step 2:
Make sure you import MatGridListModule, MatCardModule in your app.module.ts file.

These will enable Angular material grid list with card in your Angular application.

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

import { BrowserModule, provideClientHydration } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';

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

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

import { MatGridListModule } from '@angular/material/grid-list';

import { MatCardModule } from '@angular/material/card';

@NgModule({
```

```
  declarations: [
    AppComponent,
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    BrowserAnimationsModule,
    MatGridListModule,
    MatCardModule
  ],
  providers: [
    provideClientHydration()
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Step 3:

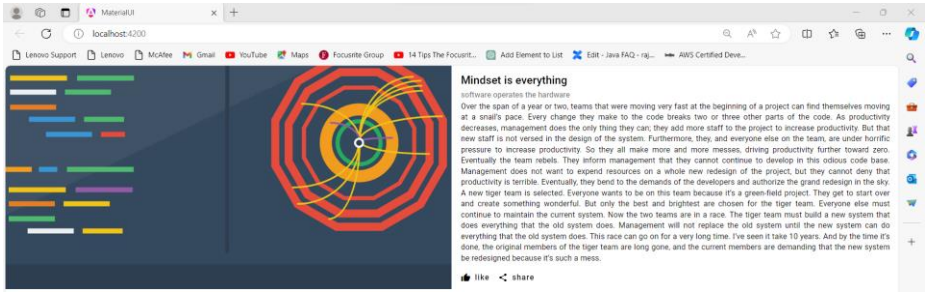
You can use Angular Material gridlist with image and card in your component template like this:

```
<mat-grid-list cols="2" rowHeight="2:1">
    <mat-grid-tile><img src="assets/banner.jpg" alt="Image"></mat-grid-tile>
    <mat-grid-tile>
        <mat-card>
            <mat-card-header>
                <mat-card-title>Mindset is everything</mat-card-title>
                <mat-card-subtitle>software operates the hardware</mat-card-subtitle>
            </mat-card-header>
            <mat-card-content style="text-align: justify">
```

Over the span of a year or two, teams that were moving very fast at the beginning of a project can find themselves moving at a snail's pace. Every change they make to the code breaks two or three other parts of the code.....etc

```
        </mat-card-content>
        <mat-card-actions>
          <button mat-button><mat-icon>thumb_up</mat-icon>like</button>
          <button mat-button><mat-icon>share</mat-icon>share</button>
        </mat-card-actions>
        <mat-card-footer></mat-card-footer>
      </mat-card>
    </mat-grid-tile>
</mat-grid-list>
```

Output:



Explanation:

In a grid we took number of columns 2 and each filled with image and card.

Instead of card we can use normal heading tags, paragraphs and buttons.

Example 11: Create Checkboxes in angular material.

Step 1:
First, make sure you have Angular Material installed in your project.

If not, you can install it using Angular CLI command

ng add @angular/material

Step 2:
Make sure you import MatCheckboxModule in your app.module.ts file.

These will enable Angular material checkboxes in your Angular application.

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

import { BrowserModule, provideClientHydration } from '@angular/platform-
browser';
```

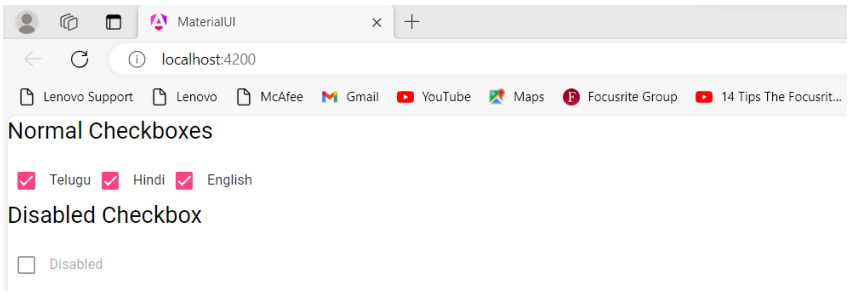
```
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { MatCheckboxModule } from '@angular/material/checkbox';
@NgModule({
  declarations: [
    AppComponent,
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    BrowserAnimationsModule,
    MatCheckboxModule
  ],
  providers: [
    provideClientHydration()
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Step 3:

You can use Angular Material checkbox in your component template like this:

```
<h1>Normal Checkboxes</h1>
<mat-checkbox>Telugu</mat-checkbox>
<mat-checkbox>Hindi</mat-checkbox>
<mat-checkbox>English</mat-checkbox>
<h1>Disabled Checkbox</h1>
<mat-checkbox disabled>Disabled</mat-checkbox>
```

Output:



Explanation:

The main component provided by MatCheckboxModule is MatCheckbox. It represents a checkbox control that users can interact with by clicking or tapping.

Checkboxes are commonly used for choices, such as enabling or disabling a feature, selecting or deselecting an item, or agreeing to terms and conditions.

Example 12: Create Radio buttons in angular material.

Step 1:
First, make sure you have Angular Material installed in your project.

If not, you can install it using Angular CLI command

ng add @angular/material

Step 2:
Make sure you import MatRadioModule in your app.module.ts file.

These will enable Angular material radio group and radio buttons in your Angular application.

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

import { BrowserModule, provideClientHydration } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';

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

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

import { MatRadioModule } from '@angular/material/radio'

@NgModule({
  declarations: [
    AppComponent,
  ],
```

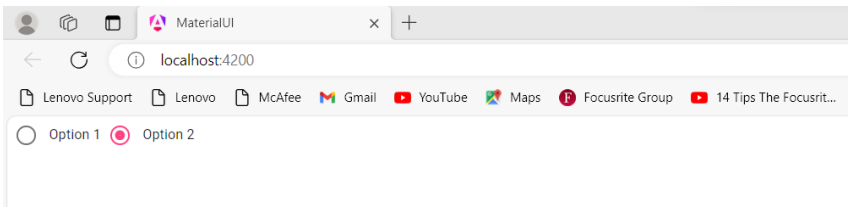
```
  imports: [
    BrowserModule,
    AppRoutingModule,
    BrowserAnimationsModule,
    MatRadioModule
  ],
  providers: [
    provideClientHydration()
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Step 3:

You can use Angular Material radio group and radio buttons in your component template like this:

```
<mat-radio-group>
    <mat-radio-button value="1">Option 1</mat-radio-button>
    <mat-radio-button value="2">Option 2</mat-radio-button>
  </mat-radio-group>
```

Output:



Explanation:

MatRadioModule is a module provided by Angular Material that facilitates the implementation of radio buttons in Angular applications.

MatRadioGroup allows you to group related radio buttons together, ensuring that only one radio button within the group can be selected at a time.

MatRadioButton Represents an individual radio button within a radio group.

Example 13: Create different select options in angular material.

Step 1:
First, make sure you have Angular Material installed in your project.

If not, you can install it using Angular CLI command

ng add @angular/material

Step 2:
Make sure you import MatSelectModule in your app.module.ts file.

These will enable Angular material select options in your Angular application.

```
import { NgModule } from '@angular/core';
import { BrowserModule, provideClientHydration } from '@angular/platform-browser';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { MatSelectModule } from '@angular/material/select';
@NgModule({
  declarations: [
    AppComponent,
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    BrowserAnimationsModule,
    MatSelectModule
  ],
  providers: [
    provideClientHydration()
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

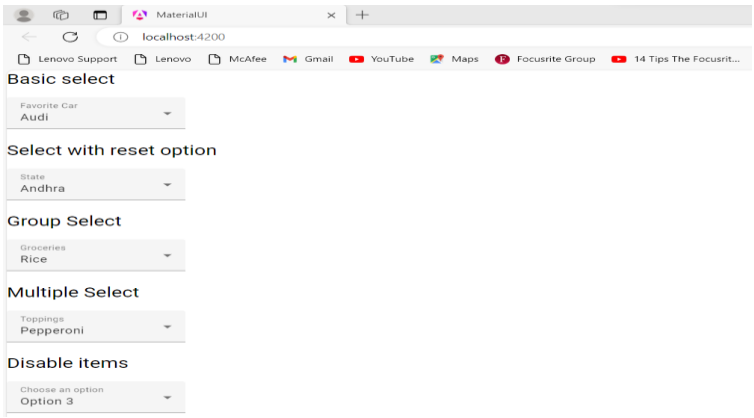
Step 3:

You can use Angular Material select options in your component template like this:

```
<h1>Basic select</h1>

<mat-form-field>
  <mat-label>Favorite Car</mat-label>
  <mat-select>
     <mat-option value="benz">Benz</mat-option>
     <mat-option value="audi">Audi</mat-option>
     <mat-option value="bmw">BMW</mat-option>
     <mat-option value="bentley">Bentley</mat-option>
  </mat-select>
</mat-form-field>

<h1>Select with reset option</h1>

<mat-form-field>
    <mat-label>State</mat-label>
    <mat-select>
      <mat-option>None</mat-option>
      <mat-option value="andhra">Andhra</mat-option>
      <mat-option value="telengana">Telangana</mat-option>
      <mat-option value="karnataka">Karnataka</mat-option>
      <mat-option value="chennai">Chennai</mat-option>
      <mat-option value="kerala">Kerala</mat-option>
    </mat-select>
  </mat-form-field>

<h1>Group Select</h1>

  <mat-form-field>
    <mat-label>Groceries</mat-label>
    <mat-select >
      <mat-option>-- None --</mat-option>
       <mat-optgroup label="Cereal">
          <mat-option value="rice">Rice</mat-option>
          <mat-option value="oats">Oats</mat-option>
          <mat-option value="wheat">Wheat</mat-option>
          <mat-option value="granola">Granola</mat-option>
       </mat-optgroup>
       <mat-optgroup label="Condiments">
          <mat-option value="salt">Salt</mat-option>
```

```
        <mat-option value="sugar">Sugar</mat-option>
      </mat-optgroup>
      <mat-optgroup label="Spices">
        <mat-option value="pepper">Pepper</mat-option>
        <mat-option value="oregano">Oregano</mat-option>
        <mat-option value="cinnamon">Cinnamon</mat-option>
      </mat-optgroup>
      <mat-optgroup label="Beverages">
        <mat-option value="milk">Milk</mat-option>
        <mat-option value="tea">Tea</mat-option>
        <mat-option value="coffee">Coffee</mat-option>
      </mat-optgroup>
    </mat-select>
  </mat-form-field>

<h1>Multiple Select</h1>

    <mat-form-field>
      <mat-label>Toppings</mat-label>
      <mat-select multiple>
        <mat-option value="extra_cheese">Extra cheese</mat-option>
        <mat-option value="mushroom">Mushroom</mat-option>
        <mat-option value="onion">Onion</mat-option>
        <mat-option value="pepperoni">Pepperoni</mat-option>
        <mat-option value="sausage">Sausage</mat-option>
        <mat-option value="tomato">Tomato</mat-option>
      </mat-select>
    </mat-form-field>

<h1>Disable items</h1>

<mat-form-field>
    <mat-label>Choose an option</mat-label>
    <mat-select>
      <mat-option value="option1">Option 1</mat-option>
      <mat-option value="option2" disabled>Option 2</mat-option>
      <mat-option value="option3">Option 3</mat-option>
    </mat-select>
  </mat-form-field>
```

Output:



Explanation:

MatSelectModule is a module provided by Angular Material that enables the usage of select dropdown components in Angular applications

The primary component provided by MatSelectModule is MatSelect. It represents a select dropdown control that allows users to choose one option from a list of options.

Select dropdowns are commonly used for presenting a list of choices in a compact and user-friendly manner.

It is designed to work inside of a <mat-form-field> element.

To add options to the select, add <mat-option> elements to the <mat-select>. Each <mat-option> has a value property that can be used to set the value that will be selected if the user chooses this option. The content of the <mat-option> is what will be shown to the user.

It is possible to disable the entire select or individual options in the select by using the disabled property on the <mat-select> and the <mat-option> elements respectively.

If you want one of your options to reset the select's value, you can omit specifying its value.

The <mat-optgroup> element can be used to group common options under a subheading. The name of the group can be set using the label property of <mat-optgroup>. Like individual <mat-option> elements, an entire <mat-optgroup> can be disabled or enabled by setting the disabled property on the group.

<mat-select> defaults to single-selection mode, but can be configured to allow multiple selection by setting the multiple property. This will allow the user to select multiple values at once. When using the <mat-select> in multiple selection mode, its value will be a sorted list of all selected values rather than a single value.

Example 14: Create different form input fields in angular material.

Step 1:
First, make sure you have Angular Material installed in your project.

If not, you can install it using Angular CLI command

ng add @angular/material

Step 2:
Make sure you import MatInputModule your app.module.ts file.

These will enable Angular material inputs in your Angular application.

MatFormFieldModule is optional to import in your Angular application.

```
import { NgModule } from '@angular/core';
import { BrowserModule, provideClientHydration } from '@angular/platform-browser';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { MatInputModule } from '@angular/material/ input;
@NgModule({
  declarations: [
    AppComponent,
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    BrowserAnimationsModule,
    MatInputModule
  ],
  providers: [
    provideClientHydration()
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

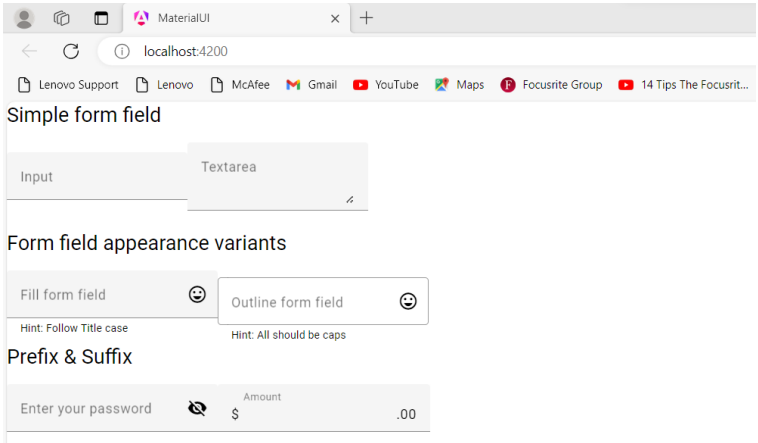
Step 3:

You can use Angular Material inputs in your component template like this:

```
<h1>Simple form field</h1>
<mat-form-field>
  <mat-label>Input</mat-label>
  <input matInput>
</mat-form-field>
<mat-form-field>
  <mat-label>Textarea</mat-label>
  <textarea matInput></textarea>
</mat-form-field>

<h1>Form field appearance variants</h1>
<mat-form-field>
  <mat-label>Fill form field</mat-label>
  <input matInput>
  <mat-icon matSuffix>sentiment_very_satisfied</mat-icon>
  <mat-hint>Hint: Follow Title case</mat-hint>
</mat-form-field>
<mat-form-field appearance="outline">
  <mat-label>Outline form field</mat-label>
  <input matInput>
  <mat-icon matSuffix>sentiment_very_satisfied</mat-icon>
  <mat-hint>Hint: All should be caps</mat-hint>
</mat-form-field>
<h1>Prefix & Suffix</h1>
<mat-form-field>
  <mat-label>Enter your password</mat-label>
  <input matInput type="password">
  <button mat-icon-button matSuffix>
```

```
    <mat-icon>visibility_off</mat-icon>
  </button>
</mat-form-field>
<mat-form-field floatLabel="always">
  <mat-label>Amount</mat-label>
  <input matInput type="number">
  <span matTextPrefix>$ </span>
  <span matTextSuffix>.00</span>
</mat-form-field>
```

Output:



Explanation:

MatFormFieldModule this module provides the <mat-form-field> component, which is a container used to wrap form controls and apply Material Design styles and animations to them.

By importing MatInputModule into your Angular application, you gain access to components and directives, allowing you to easily create styled and enhanced input fields that follow Material Design principles.

MatInput directive: This directive is applied to <input> elements to style them according to Material Design guidelines.

<textarea> element enables automatic resizing based on their content.

MatSuffix and MatPrefix directives allow you to add elements or components as suffixes or prefixes to input fields within a <mat-form-field>.

The mat-hint element is placed inside the mat-form-field. It provides a hint for the input field.

Example 15: Create different date picker fields in angular material.

Step 1:
First, make sure you have Angular Material installed in your project.

If not, you can install it using Angular CLI command

ng add @angular/material

Step 2:
Make sure you import MatInputModule your app.module.ts file.

These will enable Angular material inputs in your Angular application.

MatFormFieldModule is optional to import in your Angular application.

```
import { NgModule } from '@angular/core';
import { BrowserModule, provideClientHydration } from '@angular/platform-browser';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { MatInputModule } from '@angular/material/ input;
import { MatDatepickerModule } from '@angular/material/datepicker';
import { MatNativeDateModule } from '@angular/material/core';
@NgModule({
  declarations: [
    AppComponent,
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    BrowserAnimationsModule,
    MatInputModule,
    MatDatepickerModule,
    MatNativeDateModule
  ],
  providers: [
```

```
    provideClientHydration()
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Step 3:

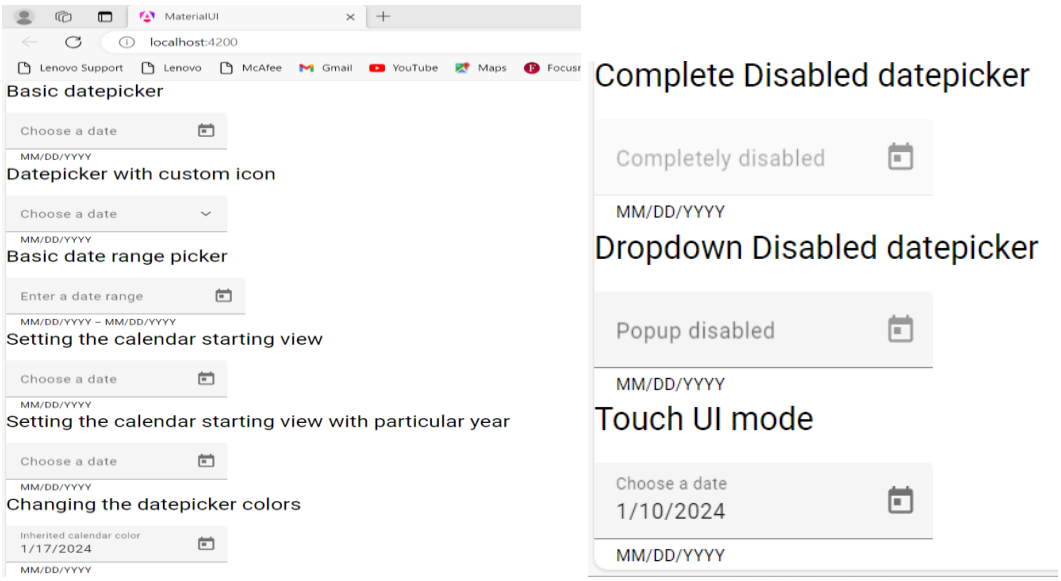
You can use Angular Material date picker in your component template like this:

```
<h1>Basic datepicker</h1>
<mat-form-field>
    <mat-label>Choose a date</mat-label>
    <input matInput [matDatepicker]="picker">
    <mat-datepicker-toggle matSuffix [for]="picker"></mat-datepicker-toggle>
    <mat-hint>MM/DD/YYYY</mat-hint>
    <mat-datepicker #picker></mat-datepicker>
</mat-form-field>

<h1>Datepicker with custom icon</h1>
<mat-form-field>
    <mat-label>Choose a date</mat-label>
    <input matInput [matDatepicker]="picker2">
    <mat-datepicker-toggle matSuffix [for]="picker2">
        <mat-icon matDatepickerToggleIcon>keyboard_arrow_down</mat-icon>
    </mat-datepicker-toggle>
    <mat-hint>MM/DD/YYYY</mat-hint>
    <mat-datepicker #picker2></mat-datepicker>
</mat-form-field>

<h1>Basic date range picker</h1>
<mat-form-field>
    <mat-label>Enter a date range</mat-label>
    <mat-date-range-input [rangePicker]="picker3">
      <input matStartDate placeholder="Start date">
```

```
      <input matEndDate placeholder="End date">
    </mat-date-range-input>
    <mat-hint>MM/DD/YYYY – MM/DD/YYYY</mat-hint>
    <mat-datepicker-toggle matIconSuffix [for]="picker3"></mat-datepicker-toggle>
    <mat-date-range-picker #picker3></mat-date-range-picker>
  </mat-form-field>

<h1>Setting the calendar starting view</h1>
<mat-form-field>
    <mat-label>Choose a date</mat-label>
    <input matInput [matDatepicker]="picker4">
    <mat-hint>MM/DD/YYYY</mat-hint>
    <mat-datepicker-toggle matIconSuffix [for]="picker4"></mat-datepicker-toggle>
    <mat-datepicker #picker4 startView="multi-year"></mat-datepicker>
  </mat-form-field>

<h1>Setting the calendar starting view with particular year</h1>
<mat-form-field>
    <mat-label>Choose a date</mat-label>
    <input matInput [matDatepicker]="picker5">
    <mat-hint>MM/DD/YYYY</mat-hint>
    <mat-datepicker-toggle matIconSuffix [for]="picker5"></mat-datepicker-toggle>
    <mat-datepicker #picker5 startView="year"></mat-datepicker>
  </mat-form-field>

 <h1>Changing the datepicker colors</h1>
  <mat-form-field color="accent">
    <mat-label>Inherited calendar color</mat-label>
    <input matInput [matDatepicker]="picker6">
    <mat-hint>MM/DD/YYYY</mat-hint>
```

```
    <mat-datepicker-toggle matIconSuffix [for]="picker6"></mat-datepicker-toggle>
    <mat-datepicker #picker6></mat-datepicker>
  </mat-form-field>

  <h1>Complete Disabled datepicker</h1>
  <mat-form-field>
    <mat-label>Completely disabled</mat-label>
    <input matInput [matDatepicker]="dp1" disabled>
    <mat-hint>MM/DD/YYYY</mat-hint>
    <mat-datepicker-toggle matIconSuffix [for]="dp1"></mat-datepicker-toggle>
    <mat-datepicker #dp1></mat-datepicker>
  </mat-form-field>

  <h1>Dropdown Disabled datepicker</h1>
  <mat-form-field>
    <mat-label>Popup disabled</mat-label>
    <input matInput [matDatepicker]="dp2">
    <mat-hint>MM/DD/YYYY</mat-hint>
    <mat-datepicker-toggle matIconSuffix [for]="dp2" disabled></mat-datepicker-
toggle>
    <mat-datepicker #dp2></mat-datepicker>
  </mat-form-field>

  <h1>Touch UI mode</h1>
  <mat-form-field>
    <mat-label>Choose a date</mat-label>
    <input matInput [matDatepicker]="picker7">
    <mat-hint>MM/DD/YYYY</mat-hint>
    <mat-datepicker-toggle matIconSuffix [for]="picker7"></mat-datepicker-toggle>
    <mat-datepicker touchUi #picker7></mat-datepicker>
  </mat-form-field>
```

Output:



Explanation:

The MatDatePicker is a component provided by Angular Material specifically designed for selecting dates. It offers a user-friendly interface for choosing dates.

The MatNativeDateModule is a module provided by Angular Material that facilitates the integration of native date functionality into Angular applications. It is specifically designed to work with Angular Material's date picker components, such as MatDatepicker.

Different browsers may have slight differences in their implementations of Date objects and date-related functionalities. MatNativeDateModule ensures compatibility with the native date handling mechanisms provided by the user's browser, helping to avoid inconsistencies or unexpected behavior across different browsers.

By default, Angular Material's date picker components work with native JavaScript Date objects. This means that the selected date values are instances of the Date class provided by the JavaScript language itself.

MatDatePicker allows users to select dates from a calendar interface. Users can navigate through months and years to pick the desired date.You can specify the format in which dates are displayed and parsed.

A datepicker is composed of a text input and a calendar pop-up, connected via the matDatepicker property on the text input.

There is also an optional datepicker toggle button that gives the user an easy way to open the datepicker pop-up.

Example 16: Create tabs in angular material.

Step 1:
First, make sure you have Angular Material installed in your project.

If not, you can install it using Angular CLI command

ng add @angular/material

Step 2:
Make sure you import MatTabsModule in your app.module.ts file.

These will enable Angular material tabs in your Angular application.

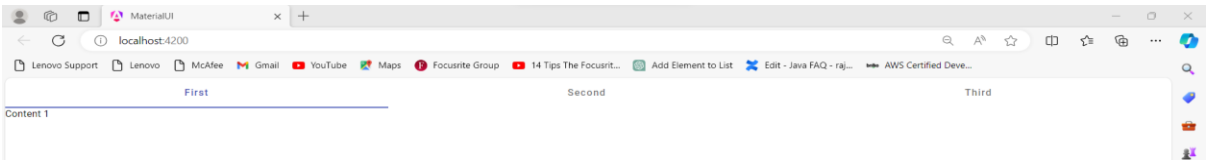
```
import { NgModule } from '@angular/core';
import { BrowserModule, provideClientHydration } from '@angular/platform-browser';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { MatTabsModule } from '@angular/material/tabs';

@NgModule({
  declarations: [
    AppComponent,
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    BrowserAnimationsModule,
    MatTabsModule
  ],
  providers: [
    provideClientHydration()
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Step 3:

You can use Angular Material tabs in your component template like this:

```
<mat-tab-group>
    <mat-tab label="First"> Content 1 </mat-tab>
    <mat-tab label="Second"> Content 2 </mat-tab>
    <mat-tab label="Third"> Content 3 </mat-tab>
</mat-tab-group>
```

Output:



Explanation:

The MatTabsModule is a module provided by Angular Material that enables the use of tab components in Angular applications. Tabs are a common UI pattern used to organize content into separate sections within the same view, allowing users to switch between them easily.

Angular Material tabs organize content into separate views where only one view can be visible at a time. Each tab's label is shown in the tab header and the active tab's label is designated with the animated ink bar. When the list of tab labels exceeds the width of the header, pagination controls appear to let the user scroll left and right across the labels.

By default, the tab group will not change its height to the height of the currently active tab. To change this, set the dynamicHeight input to true. The tab body will animate its height according to the height of the active tab.

Example 17: Create tooltip in angular material.

Step 1:
First, make sure you have Angular Material installed in your project.

If not, you can install it using Angular CLI command

```
ng add @angular/material
```

Step 2:
Make sure you import MatTooltipModule , MatButtonModule in your app.module.ts file.

These will enable Angular material tooltips to your buttons in your Angular application.

```
import { NgModule } from '@angular/core';
import { BrowserModule, provideClientHydration } from '@angular/platform-
browser';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { MatTooltipModule } from '@angular/material/tooltip';
import { MatButtonModule } from '@angular/material/button';
@NgModule({
  declarations: [
    AppComponent,
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    BrowserAnimationsModule,
    MatTooltipModule,
    MatButtonModule
  ],
  providers: [
    provideClientHydration()
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

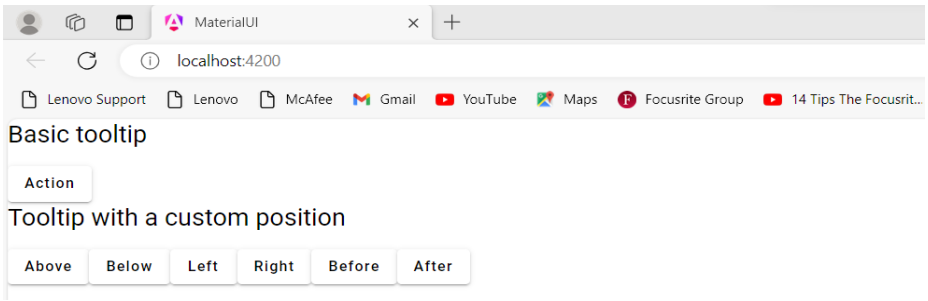
Step 3:

You can use Angular Material tooltips in your component template like this:

```
<h1>Basic tooltip</h1>
<button mat-raised-button matTooltip="hello">Action</button>

<h1>Tooltip with a custom position</h1>
```

```
<button mat-raised-button matTooltip="hello"
matTooltipPosition="above">Above</button>

<button mat-raised-button matTooltip="hello"
matTooltipPosition="below">Below</button>

<button mat-raised-button matTooltip="hello"
matTooltipPosition="left">Left</button>

<button mat-raised-button matTooltip="hello"
matTooltipPosition="right">Right</button>

<button mat-raised-button matTooltip="hello"
matTooltipPosition="before">Before</button>

<button mat-raised-button matTooltip="hello"
matTooltipPosition="after">After</button>
```

Output:



Explanation:

MatTooltipModule is a module in Angular Material that provides the matTooltip directive for creating tooltips. Tooltips are small pop-up boxes that appear when the user hovers over an element, providing additional information or context.

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

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.

Example 18: Create slide toggle in angular material.

Step 1:
First, make sure you have Angular Material installed in your project.

If not, you can install it using Angular CLI command

```
ng add @angular/material
```

Step 2:
Make sure you import MatSlideToggleModule in your app.module.ts file.

These will enable Angular material slide toggle in your Angular application.

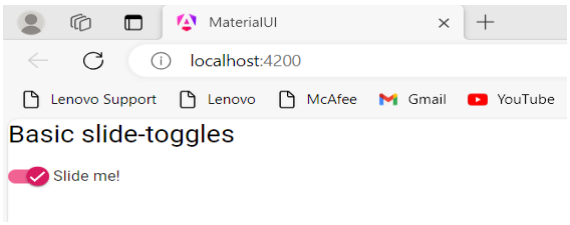
```
import { NgModule } from '@angular/core';
import { BrowserModule, provideClientHydration } from '@angular/platform-browser';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { MatSlideToggleModule } from '@angular/material/slide-toggle';
@NgModule({
  declarations: [
    AppComponent,
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    BrowserAnimationsModule,
    MatSlideToggleModule
  ],
  providers: [
    provideClientHydration()
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Step 3:

You can use Angular Material slide toggle in your component template like this:

```
<h1>Basic slide-toggles</h1>
<mat-slide-toggle>Slide me!</mat-slide-toggle>
```

Output:



Explanation:

MatSlideToggleModule is a module in Angular Material that provides functionality for creating slide toggles, which are commonly used for on/off switches in user interfaces.

The slide-toggle behaves similarly to a checkbox.

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

The color of a <mat-slide-toggle> can be changed by using the color property.

Example 19: Create expansion panel in angular material.

Step 1:
First, make sure you have Angular Material installed in your project.

If not, you can install it using Angular CLI command

ng add @angular/material

Step 2:
Make sure you import MatExpansionModule in your app.module.ts file.

These will enable Angular material expansion panel in your Angular application.

```
import { NgModule } from '@angular/core';
import { BrowserModule, provideClientHydration } from '@angular/platform-browser';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { MatExpansionModule } from '@angular/material/expansion';
@NgModule({
  declarations: [
    AppComponent,
  ],
```

```
  imports: [
    BrowserModule,
    AppRoutingModule,
    BrowserAnimationsModule,
    MatExpansionModule
  ],
  providers: [
    provideClientHydration()
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

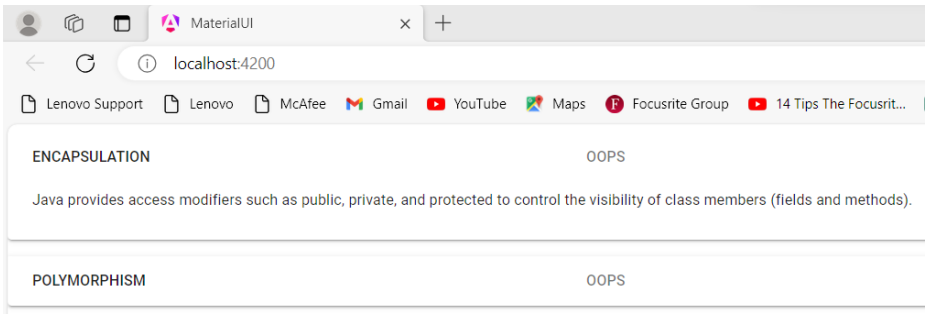
Step 3:

You can use Angular Material expansion module in your component template like this:

```
<mat-accordion>
    <mat-expansion-panel hideToggle>
      <mat-expansion-panel-header>
        <mat-panel-title>
          ENCAPSULATION
        </mat-panel-title>
        <mat-panel-description>
            OOPS
        </mat-panel-description>
      </mat-expansion-panel-header>
      <p>Java provides access modifiers such as public, private, and protected to control the visibility of class members (fields and methods).</p>
    </mat-expansion-panel>
    <mat-expansion-panel>
      <mat-expansion-panel-header>
```

```
        <mat-panel-title>
          POLYMORPHISM
        </mat-panel-title>
        <mat-panel-description>
          OOPS
        </mat-panel-description>
      </mat-expansion-panel-header>
      <p>polymorphism is mainly achieved through method overriding and interfaces. There are two types of polymorphism: compile-time (or static) polymorphism and runtime (or dynamic) polymorphism.</p>
    </mat-expansion-panel>
  </mat-accordion>
```

Output:



Explanation:

To use the expansion panel components, you need to import the MatExpansionModule in your Angular module.

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

The <mat-expansion-panel-header> shows a summary of the panel content and acts as the control for expanding and collapsing. This header may optionally contain an <mat-panel-title> and an <mat-panel-description>, which format the content of the header to align with Material Design specifications.

By default, the expansion-panel header includes a toggle icon at the end of the header to indicate the expansion state. This icon can be hidden via the hideToggle property.

Example 20: Create slider in angular material.

Step 1:
First, make sure you have Angular Material installed in your project.

If not, you can install it using Angular CLI command

ng add @angular/material

Step 2:
Make sure you import MatSliderModule in your app.module.ts file.

These will enable Angular material slider in your Angular application.

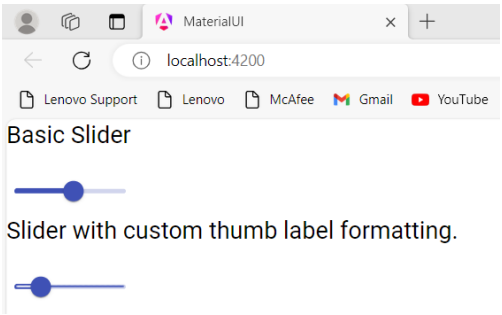
```
import { NgModule } from '@angular/core';
import { BrowserModule, provideClientHydration } from '@angular/platform-browser';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { MatSliderModule } from '@angular/material/slider';
@NgModule({
  declarations: [
    AppComponent,
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    BrowserAnimationsModule,
    MatSliderModule
  ],
  providers: [
    provideClientHydration()
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Step 3:

You can use Angular Material slider in your component template like this:

```
<h1>Basic Slider</h1>
<mat-slider>
  <input matSliderThumb>
</mat-slider>
<h1>Slider with custom thumb label formatting.</h1>
<mat-slider min="0" max="100000" step="1000" showTickMarks discrete>
  <input matSliderThumb>
</mat-slider>
```

Output:



Explanation:

MatSliderModule is a module in Angular Material that provides the mat-slider component, which is used to create sliders or range input controls in Angular applications. Sliders are commonly used to select a value within a specified range by dragging a handle along a track.

Example 21: Create Toolbar and menu in angular material.

Step 1:
First, make sure you have Angular Material installed in your project.

If not, you can install it using Angular CLI command

ng add @angular/material

Step 2:
Make sure you import MatMenuModule, MatButtonModule, MatToolbarModule, MatIconModule in your app.module.ts file.

These will enable Angular material toolbar, menu with icon buttons in your Angular application.

```
import { NgModule } from '@angular/core';
import { BrowserModule, provideClientHydration } from '@angular/platform-browser';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { MatMenuModule } from '@angular/material/menu';
import { MatButtonModule } from '@angular/material/button';
import { MatToolbarModule } from '@angular/material/toolbar';
import { MatIconModule } from '@angular/material/icon';
@NgModule({
  declarations: [
    AppComponent,
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    BrowserAnimationsModule,
    MatMenuModule,
    MatButtonModule,
    MatToolbarModule,
    MatIconModule
  ],
  providers: [
    provideClientHydration()
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Step 3:

You can use Angular Material toolbar, menu in your component template like this:

Step 3.1: Create home, about, service1 and projects components.

Step 3.2: Creates routes for above components in app-routing.module.ts file.

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { HomeComponent } from './home/home.component';
import { AboutComponent } from './about/about.component';
import { Service1Component } from './service1/service1.component';
import { ProjectsComponent } from './projects/projects.component';

const routes: Routes = [
  {path:'home', component:HomeComponent},
  {path:'about', component:AboutComponent},
  {path:'service1', component:Service1Component},
  {path:'projects', component:ProjectsComponent}
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

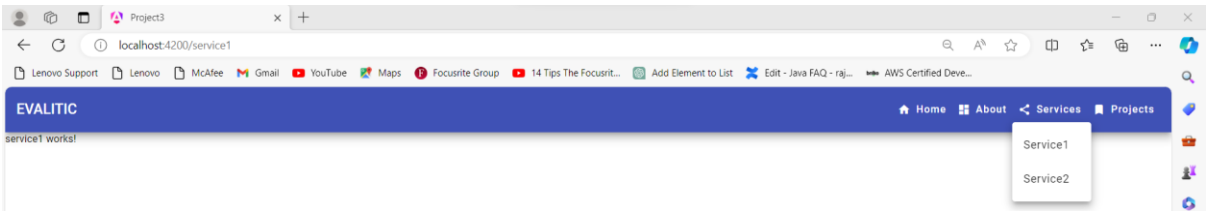
Step 3.3: Create menubar component.

Step 3.4: Add following code for menu and toolbar functionality.

```
<mat-toolbar color="primary" class="mat-elevation-z8">
  <span>EVALITIC</span>
  <span style="flex: 1 1 auto;"></span>
  <button mat-button routerLink="home">
    <mat-icon>home</mat-icon>
    Home
```

```
        </button>
        <button mat-button routerLink="about">
          <mat-icon>dashboard</mat-icon>
          About
        </button>
        <button mat-button [mat-menu-trigger-for]="services_submenu">
          <mat-icon>share</mat-icon>
          Services
        </button>
        <button mat-button routerLink="projects">
          <mat-icon>bookmark</mat-icon>
          Projects
        </button>

        <mat-menu #services_submenu="matMenu">
          <button mat-menu-item routerLink="service1">Service1</button>
          <button mat-menu-item>Service2</button>
        </mat-menu>
      </mat-toolbar>
```

Output:



Explanation:

The MatToolbarModule provides the toolbar component in Angular Material.

The toolbar is a container that holds a set of UI elements, typically used for navigation, actions, or branding within an application.

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

The Material Design specifications describe that toolbars can also have multiple rows. Creating toolbars with multiple rows in Angular Material can be done by placing

<mat-toolbar-row> elements inside of a <mat-toolbar>.Placing content outside of a <mat-toolbar-row> when multiple rows are specified is not supported.

The MatMenuModule is another module provided by Angular Material, and it is specifically related to the menu component. Angular Material's menu component allows you to create menus and context menus within your Angular applications.

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

By itself, the <mat-menu> element does not render anything. The menu is attached to and opened via application of the matMenuTriggerFor directive.

Menus support displaying mat-icon elements before the menu item text.

Material supports the ability for an mat-menu-item to open a sub-menu. To do so, you have to define your root menu and sub-menus, in addition to setting the [matMenuTriggerFor] on the mat-menu-item that should trigger the sub-menu.

routerLink is a directive used for navigation between routes. It is commonly used in conjunction with the Angular Router to navigate between different components based on the route configuration.

Remember that for routerLink to work, you need to have the Angular Router configured in your application. You should also have the necessary routes defined in your route configuration.

Example 22: Create Sidenav and menu in angular material.

Step 1:
First, make sure you have Angular Material installed in your project.

If not, you can install it using Angular CLI command

ng add @angular/material

Step 2:
Make sure you import MatSidenavModule, MatListModule, MatMenuModule, MatButtonModule, MatToolbarModule, MatIconModule in your app.module.ts file.

These will enable Angular material toolbar, menu with icon buttons in your Angular application.

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

import { BrowserModule, provideClientHydration } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';

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

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

```
import { MatMenuModule } from '@angular/material/menu';
import { MatButtonModule } from '@angular/material/button';
import { MatToolbarModule } from '@angular/material/toolbar';
import { MatIconModule } from '@angular/material/icon';
import { MatSidenavModule } from '@angular/material/sidenav';
import { MatListModule } from '@angular/material/list';
@NgModule({
  declarations: [
    AppComponent,
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    BrowserAnimationsModule,
    MatMenuModule,
    MatButtonModule,
    MatToolbarModule,
    MatIconModule,
    MatSidenavModule,
    MatListModule
  ],
  providers: [
    provideClientHydration()
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Step 3:

You can use Angular Material sidenav, toolbar, menu in your component template like this:

Step 3.1: Create home, about, service1 and projects components for toolbar menu.

Step 3.2: Create bashboard, settings, help & analytics components for sidenav menu.

Step 3.3: Creates routes for above components in app-routing.module.ts file.

```
import { Component, NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { HomeComponent } from './home/home.component';
import { AboutComponent } from './about/about.component';
import { Service1Component } from './service1/service1.component';
import { ProjectsComponent } from './projects/projects.component';
import { DashboardComponent } from './dashboard/dashboard.component';
import { SettingsComponent } from './settings/settings.component';
import { HelpmoduleComponent } from
'./helpmodule/helpmodule.component';
import { AnalyticsComponent } from './analytics/analytics.component';

const routes: Routes = [
  {path:'home', component:HomeComponent},
  {path:'about', component:AboutComponent},
  {path:'service1', component:Service1Component},
  {path:'projects', component:ProjectsComponent},
  {path:'dashboard', component:DashboardComponent},
  {path:'settings', component:SettingsComponent},
  {path:'help', component:HelpmoduleComponent},
  {path:'analytics', component:AnalyticsComponent}
];
@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Step 3.3: Create sidenavbar component.

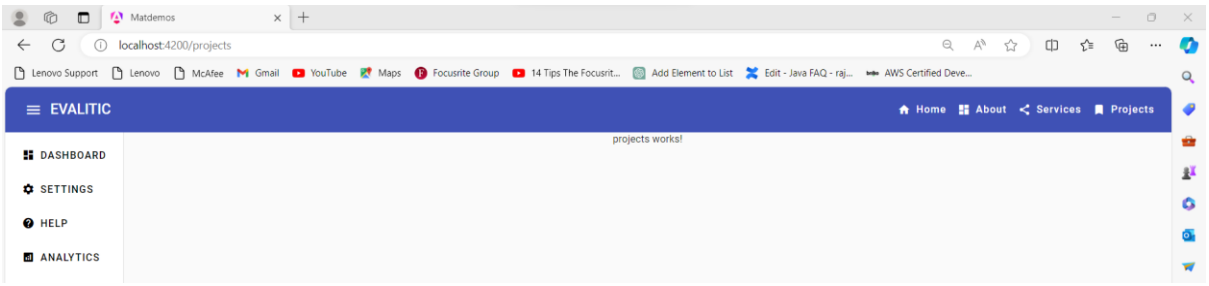
Step 3.4: Add following code for toolbar menu and sidenav menu functionality in sidenavbar component.html file.

```
<mat-toolbar color="primary" class="mat-elevation-z8">
  <button mat-icon-button (click)="drawer.toggle()"><mat-icon>menu</mat-icon></button>
  <span>EVALITIC</span>
  <span style="flex: 1 1 auto;"></span>
  <button mat-button routerLink="home">
    <mat-icon>home</mat-icon>
    Home
  </button>
  <button mat-button routerLink="about">
    <mat-icon>dashboard</mat-icon>
    About
  </button>
  <button mat-button [mat-menu-trigger-for]="services_submenu">
    <mat-icon>share</mat-icon>
    Services
  </button>
  <button mat-button routerLink="projects">
    <mat-icon>bookmark</mat-icon>
    Projects
  </button>

  <mat-menu #services_submenu="matMenu">
    <button mat-menu-item routerLink="service1">Service1</button>
    <button mat-menu-item>Service2</button>
  </mat-menu>
```

```
</mat-toolbar>

<mat-drawer-container autosize>
  <mat-drawer #drawer opened="true" mode="side" position="start">
    <mat-nav-list>
      <mat-list-item>
        <button mat-button routerLink="dashboard">
          <mat-icon>dashboard</mat-icon>
          DASHBOARD
        </button>
      </mat-list-item>
      <mat-list-item>
        <button mat-button routerLink="settings">
          <mat-icon>settings</mat-icon>
          SETTINGS
        </button>
      </mat-list-item>
      <mat-list-item>
        <button mat-button routerLink="help">
          <mat-icon>help</mat-icon>
          HELP
        </button>
      </mat-list-item>
      <mat-list-item>
        <button mat-button routerLink="analytics">
          <mat-icon>analytics</mat-icon>
          ANALYTICS
        </button>
      </mat-list-item>
    </mat-nav-list>
```

```
    </mat-drawer>
    <mat-drawer-content>
      <div style="text-align: center; min-height: 600px;">
        <router-outlet></router-outlet>
      </div>
    </mat-drawer-content>
</mat-drawer-container>
```

Output:



Explanation:

Make sure to import MatSidenavModule in the module where you want to use it.

Use the <mat-sidenav> element to create a side navigation container. You can place the content you want to display in the sidenav within this element.

You can configure the behavior and appearance of the sidenav using various attributes and properties. For example, you can use the mode attribute to set whether the sidenav should be fixed, push the content, or overlay the content.

In this example, mode="side" sets the sidenav to be a fixed side panel, and opened specifies that the sidenav should be open by default.

You can further customize the sidenav by using features provided by Angular Material, such as adding a backdrop, handling events, and more. Check the official Angular Material documentation for MatSidenav for detailed information and examples

What Next?

Here some other components like Table, Chips, Dialog, Autocomplete, Bottom Sheet, paginater, progress spinner, Ripples, Snackbar, Stepper and Tree in Angular Material UI which are interacted with binding concepts. So those will discuss in later chapters.