1. Date picker

app.component.html

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

app.module.ts

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

---

2. Displaying Data from JSON file in table

app.component.html

```
<table mat-table [dataSource]="dataSource" class="mat-elevation-z8">
    <!--- Note that these columns can be defined in any order.
          The actual rendered columns are set as a property on the row definition" -->
    <!-- Position Column -->
```

```
   <ng-container matColumnDef="position">
     <th mat-header-cell *matHeaderCellDef> No. </th>
     <td mat-cell *matCellDef="let element"> {{element.position}} </td>
   </ng-container>
   <!-- Name Column -->
   <ng-container matColumnDef="name">
     <th mat-header-cell *matHeaderCellDef> Name </th>
     <td mat-cell *matCellDef="let element"> {{element.name}} </td>
   </ng-container>
   <!-- Weight Column -->
   <ng-container matColumnDef="weight">
     <th mat-header-cell *matHeaderCellDef> Weight </th>
     <td mat-cell *matCellDef="let element"> {{element.weight}} </td>
   </ng-container>
   <!-- Symbol Column -->
   <ng-container matColumnDef="symbol">
     <th mat-header-cell *matHeaderCellDef> Symbol </th>
     <td mat-cell *matCellDef="let element"> {{element.symbol}} </td>
   </ng-container>
   <tr mat-header-row *matHeaderRowDef="displayedColumns"></tr>
   <tr mat-row *matRowDef="let row; columns: displayedColumns;"></tr>
 </table>
      <!-- Copyright 2022 Google LLC. All Rights Reserved.
     Use of this source code is governed by an MIT-style license that
     can be found in the LICENSE file at https://angular.io/license -->
```

app.component.css

```
table {
   width: 100%;
 }
```

app.component.ts

```
import { Component } from '@angular/core';
export interface PeriodicElement {
  name: string;
  position: number;
  weight: number;
  symbol: string;
}
const ELEMENT_DATA: PeriodicElement[] = [
  {position: 1, name: 'Hydrogen', weight: 1.0079, symbol: 'H'},
  {position: 2, name: 'Helium', weight: 4.0026, symbol: 'He'},
  {position: 3, name: 'Lithium', weight: 6.941, symbol: 'Li'},
  {position: 4, name: 'Beryllium', weight: 9.0122, symbol: 'Be'},
  {position: 5, name: 'Boron', weight: 10.811, symbol: 'B'},
  {position: 6, name: 'Carbon', weight: 12.0107, symbol: 'C'},
  {position: 7, name: 'Nitrogen', weight: 14.0067, symbol: 'N'},
  {position: 8, name: 'Oxygen', weight: 15.9994, symbol: 'O'},
  {position: 9, name: 'Fluorine', weight: 18.9984, symbol: 'F'},
  {position: 10, name: 'Neon', weight: 20.1797, symbol: 'Ne'},
];
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'table';
  displayedColumns: string[] = ['position', 'name', 'weight', 'symbol'];
  dataSource = ELEMENT_DATA;
}
```

app.module.tapp.module.ts

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

3.Table-Pagination

app.component.html

```
<div class="mat-elevation-z8">
  <table mat-table [dataSource]="dataSource">
```

```
    <!-- Position Column -->
    <ng-container matColumnDef="position">
      <th mat-header-cell *matHeaderCellDef> No. </th>
      <td mat-cell *matCellDef="let element"> {{element.position}} </td>
    </ng-container>
    <!-- Name Column -->
    <ng-container matColumnDef="name">
      <th mat-header-cell *matHeaderCellDef> Name </th>
      <td mat-cell *matCellDef="let element"> {{element.name}} </td>
    </ng-container>
    <!-- Weight Column -->
    <ng-container matColumnDef="weight">
      <th mat-header-cell *matHeaderCellDef> Weight </th>
      <td mat-cell *matCellDef="let element"> {{element.weight}} </td>
    </ng-container>
    <!-- Symbol Column -->
    <ng-container matColumnDef="symbol">
      <th mat-header-cell *matHeaderCellDef> Symbol </th>
      <td mat-cell *matCellDef="let element"> {{element.symbol}} </td>
    </ng-container>
    <tr mat-header-row *matHeaderRowDef="displayedColumns"></tr>
    <tr mat-row *matRowDef="let row; columns: displayedColumns;"></tr>
  </table>
  <mat-paginator [pageSizeOptions]="[5, 10, 20]"
                 showFirstLastButtons
                 aria-label="Select page of periodic elements">
  </mat-paginator>
</div>
<!-- Copyright 2022 Google LLC. All Rights Reserved.
    Use of this source code is governed by an MIT-style license that
    can be found in the LICENSE file at https://angular.io/license -->
```

app.component.css

```
table {
    width: 100%;
  }
```

app.component.ts

```
import {AfterViewInit, Component, ViewChild} from '@angular/core';
import {MatPaginator} from '@angular/material/paginator';
import {MatTableDataSource} from '@angular/material/table';
import { DataSource } from '@angular/cdk/table';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent implements AfterViewInit{
  title = 'table-pagination';
  displayedColumns: string[] = ['position', 'name', 'weight', 'symbol'];
  dataSource = new MatTableDataSource<PeriodicElement>(ELEMENT_DATA);
  @ViewChild(MatPaginator)
  paginator!: MatPaginator;
  ngAfterViewInit() {
    this.dataSource.paginator = this.paginator;
  }
}
export interface PeriodicElement {
  name: string;
  position: number;
  weight: number;
  symbol: string;
}
const ELEMENT_DATA: PeriodicElement[] = [
  {position: 1, name: 'Hydrogen', weight: 1.0079, symbol: 'H'},
  {position: 2, name: 'Helium', weight: 4.0026, symbol: 'He'},
  {position: 3, name: 'Lithium', weight: 6.941, symbol: 'Li'},
  {position: 4, name: 'Beryllium', weight: 9.0122, symbol: 'Be'},
  {position: 5, name: 'Boron', weight: 10.811, symbol: 'B'},
  {position: 6, name: 'Carbon', weight: 12.0107, symbol: 'C'},
  {position: 7, name: 'Nitrogen', weight: 14.0067, symbol: 'N'},
  {position: 8, name: 'Oxygen', weight: 15.9994, symbol: 'O'},
  {position: 9, name: 'Fluorine', weight: 18.9984, symbol: 'F'},
  {position: 10, name: 'Neon', weight: 20.1797, symbol: 'Ne'},
  {position: 11, name: 'Sodium', weight: 22.9897, symbol: 'Na'},
  {position: 12, name: 'Magnesium', weight: 24.305, symbol: 'Mg'},
  {position: 13, name: 'Aluminum', weight: 26.9815, symbol: 'Al'},
  {position: 14, name: 'Silicon', weight: 28.0855, symbol: 'Si'},
  {position: 15, name: 'Phosphorus', weight: 30.9738, symbol: 'P'},
  {position: 16, name: 'Sulfur', weight: 32.065, symbol: 'S'},
  {position: 17, name: 'Chlorine', weight: 35.453, symbol: 'Cl'},
  {position: 18, name: 'Argon', weight: 39.948, symbol: 'Ar'},
  {position: 19, name: 'Potassium', weight: 39.0983, symbol: 'K'},
  {position: 20, name: 'Calcium', weight: 40.078, symbol: 'Ca'},
];
```

app.module.ts

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import {FormsModule, ReactiveFormsModule} from '@angular/forms';
import {MatNativeDateModule} from '@angular/material/core';
import {HttpClientModule} from '@angular/common/http';
import {MatPaginatorModule} from '@angular/material/paginator';
import {MatTableModule} from '@angular/material/table';
@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    BrowserAnimationsModule,
    FormsModule,
    ReactiveFormsModule,
    MatNativeDateModule,
    HttpClientModule,
    MatPaginatorModule,
    MatTableModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

4.Dynamic Field

app.component.html

```
<div class="container">

  <h1> Dynamically Add New Input Fields in Angular </h1>

  <form [formGroup]="productForm" (ngSubmit)="onSubmit()">

    <p>
      <label for="name">Product Name:</label>
      <input type="text" id="name" name="name" formControlName="name"
class="form-control">
    </p>

    <table class="table table-bordered" formArrayName="quantities">
      <tr>
        <th colspan="2">Add Multiple Quantity:</th>
        <th width="150px"><button type="button" (click)="addQuantity()" class="btn btn-
primary">Add More</button></th>
      </tr>
      <tr *ngFor="let quantity of quantities().controls; let i=index"
[formGroupName]="i">
        <td>
           Quantity :
           <input type="text" formControlName="qty" class="form-control">
        </td>
        <td>
           Price:
           <input type="text" formControlName="price" class="form-control">
        </td>
        <td>
           <button (click)="removeQuantity(i)" class="btn btn-
danger">Remove</button>
        </td>
      </tr>
    </table>

    <button type="submit" class="btn btn-success">Submit</button>

  </form>

  <br/>
  {{this.productForm.value | json}}
</div>
```

app.component.ts

```
import { Component } from '@angular/core';
import { FormGroup, FormControl, FormArray, FormBuilder } from '@angular/forms'

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  name = 'Angular';

  productForm: FormGroup;

  constructor(private fb:FormBuilder) {

  this.productForm = this.fb.group({
      name: '',
      quantities: this.fb.array([]) ,
    });
  }

  quantities() : FormArray {
    return this.productForm.get("quantities") as FormArray
  }

  newQuantity(): FormGroup {
    return this.fb.group({
      qty: '',
      price: '',
    })
  }

  addQuantity() {
    this.quantities().push(this.newQuantity());
  }

  removeQuantity(i:number) {
    this.quantities().removeAt(i);
  }

  onSubmit() {
    console.log(this.productForm.value);
  }
}
```

<u>app.module.ts</u>

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';

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

@NgModule({
  imports:      [ BrowserModule, FormsModule, ReactiveFormsModule ],
  declarations: [ AppComponent ],
  bootstrap:    [ AppComponent ]
})
export class AppModule { }
```

5.Mobile number

app.component.html

```
<div [formGroup]="form">
  <mat-form-field appearance="fill">
    <mat-label>Phone number</mat-label>
    <example-tel-input formControlName="tel" required></example-tel-input>
    <mat-icon matSuffix>phone</mat-icon>
    <mat-hint>Include area code</mat-hint>
  </mat-form-field>
</div>

<!-- Copyright 2022 Google LLC. All Rights Reserved.
    Use of this source code is governed by an MIT-style license that
    can be found in the LICENSE file at https://angular.io/license -->
```

app.component.css

```
.example-tel-input-container {
    display: flex;
  }

  .example-tel-input-element {
    border: none;
    background: none;
    padding:  0;
    outline: none;
    font: inherit;
    text-align: center;
  }

  .example-tel-input-spacer {
    opacity: 0;
    transition: opacity 200ms;
  }

  :host.example-floating .example-tel-input-spacer {
    opacity: 1;
  }
```

app.component.ts

```
import {FocusMonitor} from '@angular/cdk/a11y';
import {BooleanInput, coerceBooleanProperty} from '@angular/cdk/coercion';
import {
  Component,
  ElementRef,
  Inject,
  Input,
  OnDestroy,
  Optional,
  Self,
  ViewChild,
} from '@angular/core';
import  {
AbstractControl,
ControlValueAccessor,
FormBuilder,
FormControl,
FormGroup,
  NgControl,
  Validators,
} from '@angular/forms';
import {MAT_FORM_FIELD, MatFormField, MatFormFieldControl} from
'@angular/material/form-field';
import {Subject} from 'rxjs';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'mobile-number';
  form: FormGroup = new FormGroup({
    tel: new FormControl(new MyTel('', '', '')),
  });
}

export class MyTel {
  constructor(public area: string, public exchange: string, public subscriber: string) {}
}

/** Custom `MatFormFieldControl` for telephone number input. */
@Component({
  selector: 'example-tel-input',
  templateUrl: 'example-tel-input-example.html',
  styleUrls: ['example-tel-input-example.css'],
  providers: [{provide: MatFormFieldControl, useExisting: MyTelInput}],
  host: {
    '[class.example-floating]': 'shouldLabelFloat',
    '[id]': 'id',
  },
})
export class MyTelInput implements ControlValueAccessor,
MatFormFieldControl<MyTel>, OnDestroy {
  static nextId = 0;
  @ViewChild('area')
  areaInput!: HTMLInputElement;
  @ViewChild('exchange')
  exchangeInput!: HTMLInputElement;
  @ViewChild('subscriber')
  subscriberInput!: HTMLInputElement;

  parts = this._formBuilder.group({
    area: ['', [Validators.required, Validators.minLength(3), Validators.maxLength(3)]],
    exchange: ['', [Validators.required, Validators.minLength(3),
Validators.maxLength(3)]],
    subscriber: ['', [Validators.required, Validators.minLength(4),
Validators.maxLength(4)]],
  });
  stateChanges = new Subject<void>();
  focused = false;
  touched = false;
  controlType = 'example-tel-input';
  id = `example-tel-input-${MyTelInput.nextId++}`;
  onChange = (_: any) => {};
  onTouched = () => {};

  get empty() {
    const {
      value: {area, exchange, subscriber},
    } = this.parts;

    return !area && !exchange && !subscriber;
  }

  get shouldLabelFloat() {
    return this.focused || !this.empty;
  }

  @Input('aria-describedby')
  userAriaDescribedBy!: string;

  @Input()
  get placeholder(): string {
    return this._placeholder;


  }
```

```
set placeholder(value: string) {
  this._placeholder = value;
  this.stateChanges.next();
}
private _placeholder!: string;

@Input()
get required(): boolean {
  return this._required;
}
set required(value: BooleanInput) {
  this._required = coerceBooleanProperty(value);
  this.stateChanges.next();
}
private _required = false;

@Input()
get disabled(): boolean {
  return this._disabled;
}
set disabled(value: BooleanInput) {
  this._disabled = coerceBooleanProperty(value);
  this._disabled ? this.parts.disable() : this.parts.enable();
  this.stateChanges.next();
}
private _disabled = false;

@Input()
get value(): MyTel | null {
  if (this.parts.valid) {
  const {
      value: {area, exchange, subscriber},
    } = this.parts;
    return new MyTel(area!, exchange!, subscriber!);
  }
  return null;
}
set value(tel: MyTel | null) {
  const {area, exchange, subscriber} = tel || new MyTel('', '', '');
  this.parts.setValue({area, exchange, subscriber});
  this.stateChanges.next();
}

get errorState(): boolean {
  return this.parts.invalid && this.touched;


}
```

```
constructor(
  private _formBuilder: FormBuilder,
  private _focusMonitor: FocusMonitor,
  private _elementRef: ElementRef<HTMLElement>,
  @Optional() @Inject(MAT_FORM_FIELD) public _formField: MatFormField,
  @Optional() @Self() public ngControl: NgControl,
) {
  if (this.ngControl != null) {
    this.ngControl.valueAccessor = this;
  }
}

ngOnDestroy() {
  this.stateChanges.complete();
  this._focusMonitor.stopMonitoring(this._elementRef);
}

onFocusIn(event: FocusEvent) {
  if (!this.focused) {
    this.focused = true;
    this.stateChanges.next();
  }
}

onFocusOut(event: FocusEvent) {
  if (!this._elementRef.nativeElement.contains(event.relatedTarget as Element)) {
    this.touched = true;
    this.focused = false;
    this.onTouched();
    this.stateChanges.next();
  }
}

autoFocusNext(control: AbstractControl, nextElement?: HTMLInputElement): void {
  if (!control.errors && nextElement) {
    this._focusMonitor.focusVia(nextElement, 'program');
  }
}

autoFocusPrev(control: AbstractControl, prevElement: HTMLInputElement): void {
  if (control.value.length < 1) {
    this._focusMonitor.focusVia(prevElement, 'program');
  }
}
```

```
  setDescribedByIds(ids: string[]) {
    const controlElement = this._elementRef.nativeElement.querySelector(
      '.example-tel-input-container',
    )!;
    controlElement.setAttribute('aria-describedby', ids.join(' '));
  }

  onContainerClick() {
    if (this.parts.controls.subscriber.valid) {
      this._focusMonitor.focusVia(this.subscriberInput, 'program');
    } else if (this.parts.controls.exchange.valid) {
      this._focusMonitor.focusVia(this.subscriberInput,  'program');
    } else if (this.parts.controls.area.valid) {
      this._focusMonitor.focusVia(this.exchangeInput, 'program');
    } else {
      this._focusMonitor.focusVia(this.areaInput, 'program');
    }
  }

  writeValue(tel: MyTel | null): void {
    this.value = tel;
  }

  registerOnChange(fn: any): void {
    this.onChange = fn;
  }

  registerOnTouched(fn: any): void {
    this.onTouched = fn;
  }

  setDisabledState(isDisabled: boolean): void {
    this.disabled = isDisabled;
  }

  _handleInput(control: AbstractControl, nextElement?: HTMLInputElement): void {
    this.autoFocusNext(control, nextElement);
    this.onChange(this.value);
  }
}
```

app.module.ts

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent , MyTelInput} from './app.component';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import {FormsModule, ReactiveFormsModule} from '@angular/forms';
import {MatNativeDateModule} from '@angular/material/core';
import {HttpClientModule} from '@angular/common/http';
import {MatFormFieldModule} from '@angular/material/form-field';

@NgModule({
  declarations: [
  AppComponent,
  MyTelInput
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    BrowserAnimationsModule,
    FormsModule,
    ReactiveFormsModule,
    MatNativeDateModule,
    HttpClientModule,
    MatFormFieldModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

6.No White Space exists

app.component.html

```
<h1> Angular - Create Custom Validators </h1>

<form [formGroup]="form" (ngSubmit)="submit()">

    <div class="form-group">
        <label for="username">Username</label>
        <input
            formControlName="username"
            id="username"
            type="text"
            class="form-control">
```

```
      <div *ngIf="f.username.touched && f.username.invalid" class="alert alert-
danger">
         <div *ngIf="f.username.errors.required">Username is required.</div>
         <div *ngIf="f.username.errors.minlength">Username should be 3
character.</div>
         <div *ngIf="f.username.errors.cannotContainSpace">Username cannot
contain space.</div>
      </div>
   </div>

   <div class="form-group">
      <label for="password">Password</label>
      <input
         formControlName="password"
         id="password"
         type="password"
         class="form-control">
      <div *ngIf="f.password.touched && f.password.invalid" class="alert alert-
danger">
         <div *ngIf="f.password.errors.required">Password is required.</div>
      </div>
   </div>

   <button class="btn btn-primary" type="submit">Submit</button>
</form>
```

app.component.ts

```
import { Component } from '@angular/core';
import { FormGroup, FormControl, Validators} from '@angular/forms';
import { UsernameValidator } from './username.validator';

@Component({
 selector: 'app-root',
 templateUrl: './app.component.html',
 styleUrls: ['./app.component.css']
})
export class AppComponent {
 form = new FormGroup({
 username: new FormControl('', [Validators.required, Validators.minLength(3),
UsernameValidator.cannotContainSpace]),
 password: new FormControl('', Validators.required)
 });

 get f(){
 return this.form.controls;
 }

 submit(){
 console.log(this.form.value);
 }
}
```

app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';

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

@NgModule({
 declarations: [
 AppComponent
 ],
 imports: [
 BrowserModule,
 FormsModule,
 ReactiveFormsModule
 ],
 providers: [],
 bootstrap: [AppComponent]
})
export class AppModule { }
```

7.Table-Pagination

app.component.ts

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

@Component({
  selector: 'app-root',
  // templateUrl: './app.component.html',
  template: `
 <p>Screen width: {{ screenWidth }}</p>
 <p>Screen height: {{ screenHeight }}</p>
 `,
  styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit{
  title = 'height-width';


 public screenWidth: any;
 public screenHeight: any;

  ngOnInit() {
    this.screenWidth = window.innerWidth;
    this.screenHeight = window.innerHeight;
    }
}
```

8.height-width-responsive

app.component.ts

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

@Component({
  selector: 'my-app',
  template: `
     <p>Screen width: {{ screenWidth }}</p>
     <p>Screen height: {{ screenHeight }}</p>
  `,
  styleUrls: [ './app.component.css' ]
})
export class AppComponent implements OnInit {
  name = 'Angular';

  public screenWidth: any;
  public screenHeight: any;

  ngOnInit() {
      this.screenWidth = window.innerWidth;
      this.screenHeight = window.innerHeight;
  }

  @HostListener('window:resize', ['$event'])
  onResize(event) {
    this.screenWidth = window.innerWidth;
    this.screenHeight = window.innerHeight;
  }

}
```

---

9. Form

app.component.html

```
<mat-card id="card" style="text-align: center;">
  <mat-card-title>
    Form
  </mat-card-title>
<mat-form-field appearance="fill" >
  <mat-label>Input</mat-label>
  <input matInput>
</mat-form-field><br/>
<mat-form-field appearance="fill">
  <mat-label>Select</mat-label>
  <mat-select>
    <mat-option value="one">First option</mat-option>
    <mat-option value="two">Second option</mat-option>
  </mat-select>
</mat-form-field><br/>
<mat-form-field appearance="fill">
  <mat-label>Textarea</mat-label>
  <textarea matInput></textarea>
</mat-form-field><br/>
<div [formGroup]="form">
<mat-form-field appearance="fill">
  <mat-label>Phone number</mat-label>
  <example-tel-input formControlName="tel" required></example-tel-input>
  <mat-icon matSuffix>phone</mat-icon>
  <mat-hint>Include area code</mat-hint>
</mat-form-field>
</div><br/>
<label id="example-radio-group-label">Gender</label>
<mat-radio-group
  aria-labelledby="example-radio-group-label"
  class="example-radio-group"
  [(ngModel)]="favoriteSeason">
  <mat-radio-button class="example-radio-button" *ngFor="let season of seasons"
[value]="season">
    {{season}}
  </mat-radio-button>
</mat-radio-group>
<button mat-raised-button color="primary" style="margin-right:5px;"
(click)="onSave()">register</button>
</mat-card>
```

<u>app.component.css</u>

```
:host {
    display: flex;
    flex-direction: column;
    align-items: flex-start;
  }
  .example-radio-group {
    display: flex;
    flex-direction: column;
    margin: 15px 0;
    align-items: flex-start;
  }
  .example-radio-button {
    margin: 5px;
  }
```

app.component.ts

```
import {FocusMonitor} from '@angular/cdk/a11y';
import {BooleanInput, coerceBooleanProperty} from '@angular/cdk/coercion';
import {
  Component,
  ElementRef,
  Inject,
  Input,
  OnDestroy,
  Optional,
  Self,
  ViewChild,
} from '@angular/core';
import  {
AbstractControl,
ControlValueAccessor,
FormBuilder,
FormControl,
FormGroup,
  NgControl,
  Validators,
} from '@angular/forms';
import {MAT_FORM_FIELD, MatFormField, MatFormFieldControl} from
'@angular/material/form-field';
import {Subject} from 'rxjs';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'form';
  form: FormGroup = new FormGroup({
    tel: new FormControl(new MyTel('', '', '')),
  });
  favoriteSeason!: string;
  seasons: string[] = ['Male','Female'];
  onSave(){
    alert("Register Sucessfully!")
  }
}
export class MyTel {
  constructor(public area: string, public exchange: string, public subscriber: string) {}
}
@Component({
  selector: 'example-tel-input',
  templateUrl: 'example-tel-input-example.html',
  styleUrls: ['example-tel-input-example.css'],
  providers: [{provide: MatFormFieldControl, useExisting: MyTelInput}],
  host: {
    '[class.example-floating]': 'shouldLabelFloat',
    '[id]': 'id',
  },
})

export class MyTelInput implements ControlValueAccessor,
MatFormFieldControl<MyTel>, OnDestroy {
  static nextId = 0;
  @ViewChild('area')
  areaInput!: HTMLInputElement;
  @ViewChild('exchange')
  exchangeInput!: HTMLInputElement;
  @ViewChild('subscriber')
  subscriberInput!: HTMLInputElement;

  parts = this._formBuilder.group({
    area: ['', [Validators.required, Validators.minLength(3), Validators.maxLength(3)]],
    exchange: ['', [Validators.required, Validators.minLength(3),
Validators.maxLength(3)]],
    subscriber: ['', [Validators.required, Validators.minLength(4),
Validators.maxLength(4)]],
  });
  stateChanges = new Subject<void>();
  focused = false;
  touched = false;
  controlType = 'example-tel-input';
  id = `example-tel-input-${MyTelInput.nextId++}`;
  onChange = (_: any) => {};
```

```
onTouched = () => {};

get empty() {
  const {
    value: {area, exchange, subscriber},
  } = this.parts;

  return !area && !exchange && !subscriber;
}

get shouldLabelFloat() {
  return this.focused || !this.empty;
}

@Input('aria-describedby')
userAriaDescribedBy!: string;

@Input()
get placeholder(): string {
  return this._placeholder;
}
set placeholder(value: string) {
  this._placeholder = value;
  this.stateChanges.next();
}
private _placeholder!: string;

@Input()
get required(): boolean {
  return this._required;
}
set required(value: BooleanInput) {
  this._required = coerceBooleanProperty(value);
  this.stateChanges.next();
}
private _required = false;

@Input()
get disabled(): boolean {
  return this._disabled;
}
set disabled(value: BooleanInput) {
  this._disabled = coerceBooleanProperty(value);
  this._disabled ? this.parts.disable() : this.parts.enable();
  this.stateChanges.next();
}
```

```
private _disabled = false;

@Input()
get value(): MyTel | null {
  if (this.parts.valid) {
  const {
      value: {area, exchange, subscriber},
    } = this.parts;
    return new MyTel(area!, exchange!, subscriber!);
  }
  return null;
}
set value(tel: MyTel | null) {
  const {area, exchange, subscriber} = tel || new MyTel('', '', '');
  this.parts.setValue({area, exchange, subscriber});
  this.stateChanges.next();
}

get errorState(): boolean {
  return this.parts.invalid && this.touched;
}

constructor(
  private _formBuilder: FormBuilder,
  private _focusMonitor: FocusMonitor,
  private _elementRef: ElementRef<HTMLElement>,
  @Optional() @Inject(MAT_FORM_FIELD) public _formField: MatFormField,
  @Optional() @Self() public ngControl: NgControl,
) {
  if (this.ngControl != null) {
    this.ngControl.valueAccessor = this;
  }
}

ngOnDestroy() {
  this.stateChanges.complete();
  this._focusMonitor.stopMonitoring(this._elementRef);
}

onFocusIn(event: FocusEvent) {
  if (!this.focused) {
    this.focused = true;
    this.stateChanges.next();
  }
}
```

```
onFocusOut(event: FocusEvent) {
  if (!this._elementRef.nativeElement.contains(event.relatedTarget as Element)) {
    this.touched = true;
    this.focused = false;
    this.onTouched();
    this.stateChanges.next();
  }
}

autoFocusNext(control: AbstractControl, nextElement?: HTMLInputElement): void {
  if (!control.errors && nextElement) {
    this._focusMonitor.focusVia(nextElement, 'program');
  }
}

autoFocusPrev(control: AbstractControl, prevElement: HTMLInputElement): void {
  if (control.value.length < 1) {
    this._focusMonitor.focusVia(prevElement, 'program');
  }
}

setDescribedByIds(ids: string[]) {
  const controlElement = this._elementRef.nativeElement.querySelector(
    '.example-tel-input-container',
  )!;
  controlElement.setAttribute('aria-describedby', ids.join(' '));
}

onContainerClick() {
  if (this.parts.controls.subscriber.valid) {
    this._focusMonitor.focusVia(this.subscriberInput, 'program');
  } else if (this.parts.controls.exchange.valid) {
    this._focusMonitor.focusVia(this.subscriberInput, 'program');
  } else if (this.parts.controls.area.valid) {
    this._focusMonitor.focusVia(this.exchangeInput, 'program');
  } else {
    this._focusMonitor.focusVia(this.areaInput, 'program');
  }
}

writeValue(tel: MyTel | null): void {
  this.value = tel;
}

registerOnChange(fn: any): void {
  this.onChange = fn;
  }

  registerOnTouched(fn: any): void {
    this.onTouched = fn;
  }

  setDisabledState(isDisabled: boolean): void {
    this.disabled = isDisabled;
  }

  _handleInput(control: AbstractControl, nextElement?: HTMLInputElement): void {
    this.autoFocusNext(control, nextElement);
    this.onChange(this.value);
  }
}
```

app.module.ts

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent , MyTelInput} from './app.component';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import {FormsModule, ReactiveFormsModule} from '@angular/forms';
import {MatNativeDateModule} from '@angular/material/core';
import {HttpClientModule} from '@angular/common/http';
import { MatFormFieldModule } from '@angular/material/form-field';
import { MatInputModule } from '@angular/material/input';
import {MatSelectModule} from '@angular/material/select';
import {MatCardModule} from '@angular/material/card';
import {MatRadioModule} from '@angular/material/radio';
import {MatIconModule} from '@angular/material/icon';
@NgModule({
  declarations: [
    AppComponent, MyTelInput
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    BrowserAnimationsModule,
    FormsModule,
    ReactiveFormsModule,
    MatNativeDateModule,
    HttpClientModule,
    MatFormFieldModule,
    MatInputModule,
    MatSelectModule,
    MatCardModule,
    MatRadioModule,
    MatIconModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```