Directives of Angular

- Directive is a function.
- Angular directive function is responsible for converting the static DOM element into dynamic DOM element.
- Directive makes HTML more declarative.
- Directives have different types of functionalities, they can be used as
  - Element
  - Attribute
  - Class
  - Comment
- **Directives are used as Elements to return markup.**
  <ng-form> </ng-form>
  <router-outlet> </router-outlet>
- **Directives are used as Attributes to extend markup.**
  <input ngModel>
- **Directives are used as classes to make HTML more interactive and responsive.**
  ```
  <style>
  .ng-valid {
  }
  </style>
  <span class=”ng-valid”> </span>
  ```
- **Directive are used as comments to target legacy browsers**
  <!-- ngModel:Name -->
- **Angular directives are classified into 3 groups**
  - Component Directives
  - Structural Directives
  - Attribute Directives

Component Directives

- The component directive is one of the most commonly used directives in Angular.
- A component directive returns a presentation and provides an UI from where user can interact with application.
- It provides a reusable component across application.
- Angular allows to build your own component directives and also provides several pre-defined component directives.
- The pre-defined component directives are provided by using “Angular Material” library.
- You can create re-usable components.

Syntax:

Index.html

<app-login> </app-login>

Structural Directives

- A structural directive is responsible for changing the DOM structure dynamically.
- These directives allow to
    - Add a new element into DOM
    - Remove Element from DOM
    - Modify the data of Element etc.
- In JavaScript and jQuery, we use to do this with the help of functions
    -

      document.getElementById(“p”).appendChild(childElementName); JavaScript
    - $(“p”).append(childElement)
    - Lot of references are required

- Angular structural directives will make this process easy and they are:
    - **ngIF**
    - **ngSwitch**
    - **ngFor**
- Structural directives are added to HTML DOM elements to make the static element dynamic and to extend its functionality.
- They are used as attributes or properties for HTML elements.
- You can bind any structural directive by using "*"
  Syntax:
  ```
  <div  *ngIF=""> </div>
  <div *ngFor=""> </div>
  ```

Note: Every DOM element can't have definitions for multiple structural directives.

```
<div  *ngFor=""  *ngIF="">  // invalid
<div *ngFor="">
        <div *ngIF="">
        </div>
</div>
```

NgIF Directive

- It is a structural directive used to add or remove any element from DOM hierarchy dynamically.
- It uses a boolean value or condition to add or remove element.

Simple "ngIF"

- It uses a boolean value and **add element** into DOM when set to **true**.
- It uses a boolean value and **remove element** from DOM when set to **false**.

Ex:

Ifdemo.component.ts

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

@Component({
  selector: 'app-ifdemo',
  templateUrl: './ifdemo.component.html',
  styleUrls: ['./ifdemo.component.css']
})
export class IfdemoComponent{
    product = {
      Name: 'JBL Speaker',
      Price: 6000.45,
      Photo: 'assets/speaker.jpg'
    };
    isPhotoVisible = false;
    buttonText = 'Show';

    showError = false;
    userName = '';

    ToggleDisplay() {
       this.isPhotoVisible = (this.isPhotoVisible==false)?true:false;
       this.buttonText = (this.buttonText=='Show')?'Hide':'Show';
    }
```

```
     SubmitClick() {
       if(this.userName==''){
         this.showError = true;
       } else {
         this.showError = false;
         alert(`Hello ! ${this.userName}`);
       }
     }
}
```

Ifdemo.component.html

```
<div>
   <h2>Product Details</h2>
   <div class="card-deck">
      <div class="card">
         <div class="card-header">
            <h3>{{product.Name}}</h3>
            <p>{{product.Price |currency:'INR' }}</p>
         </div>
         <div class="card-body">
            <div class="form-group">
               <button (click)="ToggleDisplay()" class="btn btn-
info">{{buttonText}} Preview</button>
            </div>
            <div class="form-group">
```

```
                <img *ngIf="isPhotoVisible" [src]="product.Photo"
width="250" height="200">
            </div>
         </div>
      </div>
      <div class="card">
         <div class="card-header">
            <h3>{{product.Name}}</h3>
            <p>{{product.Price |currency:'INR' }}</p>
         </div>
         <div class="card-body">
            <div class="form-group">
               <input ngModel #preview="ngModel" name="preview"
type="checkbox"> Preview
            </div>
            <div class="form-group">
                <img *ngIf="preview.value" [src]="product.Photo"
width="250" height="200">
            </div>
         </div>
      </div>
   </div>
   <h2>Your Name</h2>
   <div class="form-inline">
```

```
        <input [(ngModel)]="userName" type="text" class="form-
control">
        <button (click)="SubmitClick()" class="btn btn-
primary">Submit</button>
    </div>
    <div *ngIf="showError">
        <span class="text-danger">User Name Required</span>
    </div>
</div>
```

Ifdemo.component.css

```
.card {
    width:300px;
}
```

Note: [ngIf] technique is used for angular dynamic containers.

***ngIf is used for HTML elements.**

Decision with alternative

- It specifies actions to perform when condition evaluates to true and another set of actions to perform when condition is false.
- "ngIf" directive provides the properties
    - then
    - else
- "then" specifies the ID of container to render when condition is true.
- "else" specifies the ID of container to render when condition is false.

Syntax:

```
<div *ngIF=”condition; then thenBlockId else elseBlockId”> </div>
```

<div ”#thenBlockId”>   // not valid

// keep the content to display when condition true;

</div>

<div ”#elseBlockId>   //not valid

// keep the content to display when condition false;

</div>

- The reference “#” is just a markup and not a dynamic reference for <div>.
- Angular provides dynamic containers to handle the interaction dynamically without effecting the DOM structure.
- Dynamic contains can respond to dynamic interactions.
- Angular provides several dynamic containers.
    - <ng-template>
    - <ng-container>
    - <ng-content>

Ex:

```
<div>
  <div *ngIf="false; then trueBlock else falseBlock"></div>
  <ng-template #trueBlock>
     Statement if  true
  </ng-template>
  <ng-template #falseBlock>
     statement if false
  </ng-template>
```

```
</div>
```

How to define an External then template?

Can we define multiple then blocks in “ngIf” directive? [else if]

- No. You can’t define multiple then blocks in “ngIf” directive.
- You can handle external then template and use for “ngIf” with the help of a directive “@ViewChild()”
- @ViewChild uses “TemplateRef<>” type.

ViewChild

- It is a property decorator.
- It configures a query in View.
- The change detector looks the first element or the directive matching the selector.
- If DOM changes and the new View Chid matches the selector then it updates with another element.
- ViewChild properties are
  - selector: The directive type or the name used for querying
  - read
  - static: True to resolve query before change detection, false to resolve query after change detection, It is default false.

Ex:

Ifdemo.component.ts

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


@Component({
  selector: 'app-ifdemo',
  templateUrl: './ifdemo.component.html',
  styleUrls: ['./ifdemo.component.css']
})
export class IfdemoComponent implements OnInit{
    isVisible = true;
    thenBlock: TemplateRef<any>|null = null;

    @ViewChild('firstBlock', {static: true}) firstBlock:
TemplateRef<any>|null = null;

    @ViewChild('secondBlock', {static: true}) secondBlock:
TemplateRef<any>|null = null;

    ngOnInit(){
      this.thenBlock = this.firstBlock;
    }
}
```

Ifdemo.component.html

```
<div>
   <div *ngIf="isVisible; then thenBlock; else elseBlock">

   </div>
   <ng-template #firstBlock>
      Then Block
```

```
  </ng-template>
  <ng-template #secondBlock>
     Another Then Block
  </ng-template>
  <ng-template #elseBlock>
      Else Block
  </ng-template>
</div>
```

Ex: Toggle Template “then block”

Ifdemo.component.ts

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

@Component({
 selector: 'app-ifdemo',
 templateUrl: './ifdemo.component.html',
 styleUrls: ['./ifdemo.component.css']
})
export class IfdemoComponent implements OnInit{
   isVisible = true;
   thenBlock: TemplateRef<any>|null = null;
```

```
  @ViewChild('firstBlock', {static: true}) firstBlock:
TemplateRef<any>|null = null;


  @ViewChild('secondBlock', {static: true}) secondBlock:
TemplateRef<any>|null = null;


  ngOnInit(){
    this.thenBlock = this.firstBlock;
  }
  ToggleBlock(){
    this.thenBlock = this.thenBlock == this.firstBlock ?
this.secondBlock : this.firstBlock;
  }
}
```

Ifdemo.component.html

```
<div>
  <div>
    <button (click)="ToggleBlock()">Toggle Then Block</button>
  </div>
  <div *ngIf="isVisible; then thenBlock else elseBlock">


  </div>
  <ng-template #firstBlock>
    Then Block
  </ng-template>
  <ng-template #secondBlock>
      Another Then Block
  </ng-template>
  <ng-template #elseBlock>
       Else Block
  </ng-template>
</div>
```

Ex: If Demo with Then block

Ifthendemo.component.ts

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

@Component({
  selector: 'app-ifthendemo',
  templateUrl: './ifthendemo.component.html',
  styleUrls: ['./ifthendemo.component.css']
})
export class IfthendemoComponent implements OnInit {

  isVisible = true;
  thenBlock: TemplateRef<any> | null = null;

  @ViewChild('details', {static: true}) details: TemplateRef<any> |
null;
```

```
  @ViewChild('preview', {static: true}) preview: TemplateRef<any> |
null;


  constructor() { }


  ngOnInit(): void {
    this.thenBlock = this.preview;
  }


  Toggle() {
    this.thenBlock = this.thenBlock == this.preview ? this.details :
this.preview;
  }
}
```

Ifthendemo.component.html

```
<div>
    <h2>Product Details</h2>
    <div class="card">
       <div class="card-header">
          <button (click)="Toggle()" class="btn btn-primary btn-
block">Details / Preview</button>
       </div>
       <div class="card-body">
         <div *ngIf="isVisible; then thenBlock"></div>
         <ng-template #details>
```

```
          <table class="table table-hover">
            <tr>
              <td>Name</td>
              <td>Nike Casuals</td>
            </tr>
            <tr>
              <td>Price</td>
              <td>6700.55</td>
            </tr>
            <tr>
              <td>Code</td>
              <td>#0101NIKE</td>
            </tr>
          </table>
        </ng-template>
        <ng-template #preview>
          <img src="assets/shoe.jpg" width="200" height="200">
        </ng-template>
      </div>
    </div>
</div>
```

The process of accessing any template content dynamically and rendering into specific location of your component is known as "Content Projection"

NgSwitch Directive

- It is a switch selector in UI.
- It can display and render only the container that is required for specific situation out of all the containers defined.
- Selector switch selects only the container to display out of a set of containers.
- Selector switch will not render the containers that are not matching with the required.
- Selector switch will Add and Remove containers from DOM.
- **Switch** block is defined by using "**ngSwitch"**
- **Case** block is defined by using **"ngSwitchCase"**
- **Default** block is defined by using "**ngSwitchDefault"**

Syntax:

```
<main-container [ngSwitch]="value/expression">
  <child-container *ngSwitchCase="1"> </child-container>
  <child-container *ngSwitchCase="2"> </child-container>
  <child-container *ngSwitchDefault> </child-container>
</main-container>
```

Ex:

Switchdemo.component.ts

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

@Component({
  selector: 'app-switchdemo',
  templateUrl: './switchdemo.component.html',
  styleUrls: ['./switchdemo.component.css']
})
export class SwitchdemoComponent {
  product = {
    Name: 'Nike Casuals',
    Price: 4600.66,
    InStock: true,
    Photo: 'assets/shoe.jpg',
    Description: 'something about nike casuals'
  };
  selectedView = 'details';
  Show(e){
     this.selectedView = e.target.name;
  }
}
```

Switchdemo.component.html

```
<div>
    <div class="btn-toolbar bg-danger">
        <div class="btn-group">
            <button name="details" (click)="Show($event)" class="btn
btn-danger">Details</button>
            <button name="preview" (click)="Show($event)" class="btn
btn-danger">Preview</button>
```

```
          <button name="description" (click)="Show($event)"
class="btn btn-danger">Description</button>
        </div>
    </div>
    <div [ngSwitch]="selectedView">
        <div class="card" *ngSwitchCase="'details'">
            <div class="card-header">
                <h2>{{product.Name}}</h2>
            </div>
            <div class="card-body">
                <h4>{{product.Price}}</h4>
            </div>
            <div class="card-footer">
                <h3>Status: {{(product.InStock==true?"Available":"Out of
Stock")}}</h3>
            </div>
        </div>
        <div class="card" *ngSwitchCase="'preview'">
            <div class="card-body">
                <img [src]="product.Photo" width="200" height="200" >
            </div>
        </div>
        <div class="card" *ngSwitchCase="'description'">
            <div class="card-body">
            <p>{{product.Description}}</p>
          </div>
        </div>
      </div>
    </div>
```
Ex:

Switchdemo.component.ts

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

@Component({
  selector: 'app-switchdemo',
  templateUrl: './switchdemo.component.html',
  styleUrls: ['./switchdemo.component.css']
})
export class SwitchdemoComponent {
  product = {
    Name: 'Nike Casuals',
    Price: 4600.66,
    InStock: true,
    Photo: 'assets/shoe.jpg',
    Description: 'something about nike casuals'
  };
  selectedView = 'details';
  views = ['details', 'preview', 'description'];
  count = 0;
  Show(e){
     this.selectedView = e.target.name;
  }
  NextClick() {
    this.count ++;
    this.selectedView = this.views[this.count];
    if(this.count == this.views.length) {
      this.selectedView = this.views[0];
    }
  }
  PreviousClick(){
    this.count --;
    this.selectedView = this.views[this.count];
  }
}
```

Switchdemo.component.html

```
<div>
    <h2>Product Details</h2>
    <div class="row">
        <div class="col-3">
            <div class="btn-toolbar bg-danger">
                <div class="btn-group btn-group-vertical">
```

```
          <button name="details" (click)="Show($event)"
class="btn btn-danger btn-block">Details</button>
          <button name="preview" (click)="Show($event)"
class="btn btn-danger btn-block">Preview</button>
          <button name="description" (click)="Show($event)"
class="btn btn-danger btn-block">Description</button>
        </div>
      </div>
    </div>
    <div class="col-9">
      <div>
        <button (click)="PreviousClick()" class="btn btn-danger">
          <span class="fa fa-backward"></span>
        </button>
        <button (click)="NextClick()" class="btn btn-danger">
          <span class="fa fa-forward"></span>
        </button>
      </div>
      <div [ngSwitch]="selectedView">
        <div class="card" *ngSwitchCase="'details'">
          <div class="card-header">
            <h2>{{product.Name}}</h2>
          </div>
          <div class="card-body">
            <h4>{{product.Price}}</h4>
          </div>
          <div class="card-footer">
            <h3>Status: {{(product.InStock==true?"Available":"Out
of Stock")}}</h3>
          </div>
        </div>
        <div class="card" *ngSwitchCase="'preview'">
          <div class="card-body">
            <img [src]="product.Photo" width="200" height="200"
>
          </div>
        </div>
        <div class="card" *ngSwitchCase="'description'">
          <div class="card-body">
            <p>{{product.Description}}</p>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
```

NgFor

- It is a repeater.
- It is used to repeat any HTML dynamically based on an iterator.
- It requires iterator over values in collection.

- It uses “of” operator to read values.
  Syntax:
  ```
  <div *ngFor=”let value of collection”>
  </div>
  ```

EX:

Fordemo.component.ts

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


@Component({
  selector: 'app-fordemo',
  templateUrl: './fordemo.component.html',
  styleUrls: ['./fordemo.component.css']
})
export class FordemoComponent{
    navItems = ['Home', 'Electronics', 'Footwear', 'Fashion', 'All'];
}
```

Fordemo.component.html

```
<div>
    <div class="row">
        <div class="col-2">
            <h3>Nav Bar</h3>
            <ul>
              <li *ngFor="let item of navItems"> <a href="#">{{item}}</a>
</li>
            </ul>
```

```
</div>
<div class="col-2">
   <h3>Nav Menu</h3>
   <select class="form-control">
      <option *ngFor="let item of navItems">
         {{item}}
      </option>
   </select>
</div>
<div class="col-2">
   <h3>Nav Table</h3>
   <table class="table table-hover">
      <tbody>
         <tr *ngFor="let item of navItems">
            <td>{{item}}</td>
         </tr>
      </tbody>
   </table>
</div>
<div class="col-2">
   <h3>Nav List</h3>
   <select size="3" class="form-control">
      <option *ngFor="let item of navItems">
         {{item}}
```

```
            </option>
          </select>
        </div>
        <div class="col-2">
          <h3>Check List</h3>
          <div class="check-list">
            <div *ngFor="let item of navItems">
              <input type="checkbox"> {{item}}
            </div>
          </div>
        </div>
    </div>
</div>
```

Fordemo.component.css

```
.check-list {
    height: 90px;
    padding: 10px;
    overflow:auto;
}
```

Ex: Nested NgFor

Fordemo.component.ts

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

@Component({
  selector: 'app-fordemo',
  templateUrl: './fordemo.component.html',
  styleUrls: ['./fordemo.component.css']
})
export class FordemoComponent{
    navItems = [
      {Category: 'Electronics', Products: ['JBL Speaker', 'Earpods']},
      {Category: 'Footwear', Products: ['Nike Casuals', 'Lee Cooper
Boot']},
      {Category: 'Fashion', Products: ['Shirt', 'Jeans']}
    ];
}
```

Fordemo.component.html

```
<div>
    <div class="row">
        <div class="col-3">
            <h3>Nav Bar</h3>
            <ul>
              <li *ngFor="let item of navItems">
                  <a href="#">{{item.Category}}</a>
                  <ul>
                      <li *ngFor="let product of item.Products">
                         {{product}}
                      </li>
              </ul>
            </li>
          </ul>
        </div>
        <div class="col-3">
          <h3>Nav Menu</h3>
          <select class="form-control">
            <optgroup *ngFor="let item of navItems"
label="{{item.Category}}" >
              <option *ngFor="let product of item.Products">
                {{product}}
              </option>
            </optgroup>
          </select>
        </div>
        <div class="col-3">
          <h3>Nav Collapse</h3>
          <div *ngFor="let item of navItems">
            <details>
              <summary>{{item.Category}}</summary>
              <ul>
                <li *ngFor="let product of item.Products">
                  <a>{{product}}</a>
                </li>
              </ul>
            </details>
          </div>
        </div>
      </div>
</div>
```

NgFor Properties

Property	Type	Description
index	number	It returns the iterator index number. You can identify any element location in iterator by using index.
first	boolean	It returns true if iterating item is the first item.
last	boolean	It returns true if iterating item is the last item.
even	boolean	It returns true if iterating item is at even occurrence.
odd	boolean	It returns true if iterating item is at odd occurrence.
trackBy	function	It is a function pointer, It uses a call back function that identifies the changes in iterator.

Syntax:

```
<li  *ngFor=”let item of collection; let i=index; let e=even”>
```

Ex:

Shopping.component.ts

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

```
@Component({
  selector: 'app-shopping',
  templateUrl: './shopping.component.html',
  styles: [
  ]
})
export class ShoppingComponent{
  categories = ['Select a Category', 'Electronics', 'Footwear', 'Fashion'];
  electronics = ['Select Electronics', 'JBL Speaker', 'Earpods'];
  footwear = ['Select Footwear', 'Nike Casuals', 'Lee Cooper Boot'];
  fashion = ['Select Fashion', 'Jeans', 'Shirt'];
  products = [];
  data = [
    {Name: 'JBL Speaker', Price: 4500.55, Photo: 'assets/speaker.jpg'},
    {Name: 'Earpods', Price: 2500.55, Photo: 'assets/earpods.jpg'},
    {Name: 'Nike Casuals', Price: 6500.55, Photo: 'assets/shoe.jpg'},
    {Name: 'Lee Cooper Boot', Price: 2500.55, Photo:
'assets/shoe1.jpg'},
    {Name: 'Jeans', Price: 1500.55, Photo: 'assets/jeans.jpg'},
    {Name: 'Shirt', Price: 2500.55, Photo: 'assets/shirt.jpg'},
  ];


  selectedCategoryName = 'Select a Category';
```

```
selectedProductName;
searchedProduct = {
  Name: '',
  Price: 0,
  Photo: ''
};
cartItems = [];
cartItemsCount = 0;
isCartVisible = false;
GetCount(){
  this.cartItemsCount = this.cartItems.length;
}

onCategoryChange(){
   switch(this.selectedCategoryName){
     case 'Electronics':
     this.products = this.electronics;
     break;
     case 'Footwear':
     this.products = this.footwear;
     break;
     case 'Fashion':
     this.products = this.fashion;
     break;
```

```
      default:
      this.products = ['Select any Category'];
      break;
    }
  }
  onProductChange(){
    this.searchedProduct =
this.data.find(x=>x.Name==this.selectedProductName);
  }
  onAddToCartClick(){
   this.cartItems.push(this.searchedProduct);
   alert('Item Added to Cart');
   this.GetCount();
  }
  onToggleCart(){
   this.isCartVisible = this.isCartVisible==false?true:false;
  }
  onRemoveClick(index){
    let flag = confirm('Are you sure? want to delete?');
    if(flag==true) {
      this.cartItems.splice(index,1);
      alert('Item Deleted from Cart');
    }
  }
```

```
}
```

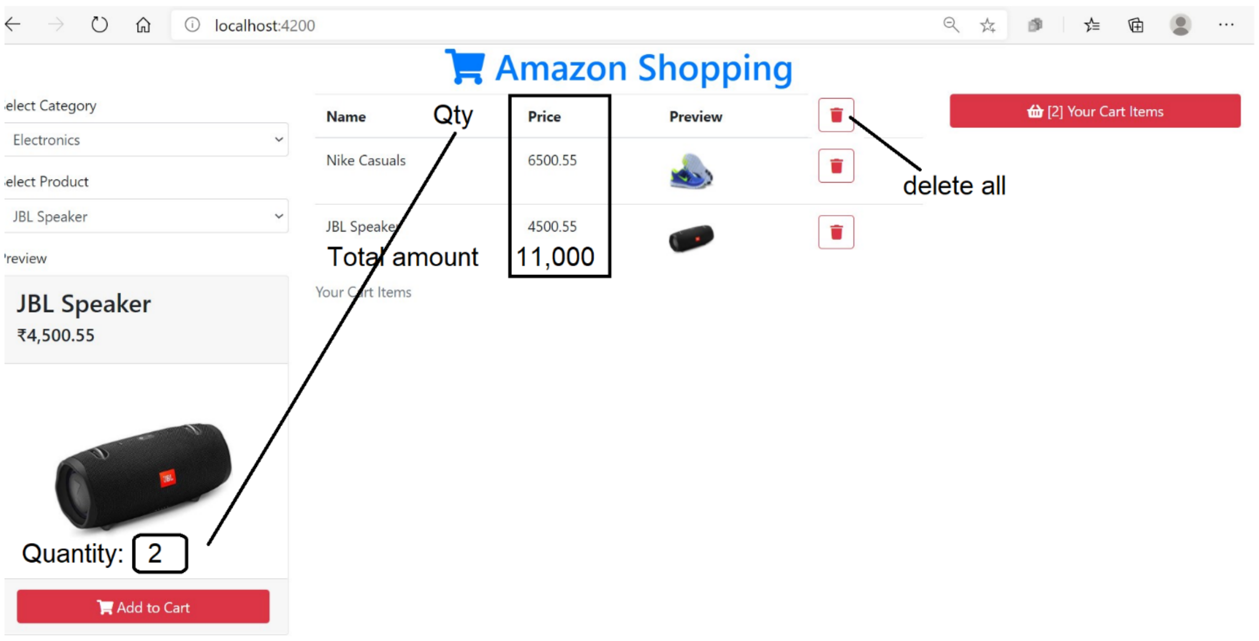
Shopping.component.html

```
<div>
   <h1 class="text-center text-primary"> <span class="fa fa-shopping-
cart"></span> Amazon Shopping </h1>
   <div class="row">
       <div class="col-3">
          <div class="form-group">
             <label>Select Category</label>
             <div>
                <select (change)="onCategoryChange()"
[(ngModel)]="selectedCategoryName" class="form-control">
                   <option *ngFor="let item of categories">
                      {{item}}
                   </option>
                </select>
             </div>
          </div>
          <div class="form-group">
             <label>Select Product</label>
             <div>
                <select (change)="onProductChange()"
[(ngModel)]="selectedProductName" class="form-control">
                   <option *ngFor="let item of products">
                      {{item}}
```

```
            </option>
          </select>
        </div>
      </div>
      <div class="form-group">
        <label>Preview</label>
        <div class="card">
          <div class="card-header">
            <h3>{{searchedProduct.Name}}</h3>
            <h5>{{searchedProduct.Price | currency:'INR'}}</h5>
          </div>
          <div class="card-body text-center">
            <img [src]="searchedProduct.Photo" width="200"
height="200">
          </div>
          <div class="card-footer text-center">
            <button (click)="onAddToCartClick()" class="btn btn-
danger btn-block">
              <span class="fa fa-shopping-cart"></span>
               Add to Cart
            </button>
          </div>
        </div>
      </div>
```

```
    </div>
    <div class="col-6">
      <table *ngIf="isCartVisible" class="table table-hover">
        <caption>Your Cart Items</caption>
        <thead>
          <tr>
            <th>Name</th>
            <th>Price</th>
            <th>Preview</th>
          </tr>
        </thead>
        <tbody>
          <tr *ngFor="let item of cartItems; let i=index">
            <td>{{item.Name}}</td>
            <td>{{item.Price}}</td>
            <td><img [src]="item.Photo" width="50"
height="50"></td>
            <td>
              <button (click)="onRemoveClick(i)" class="btn btn-
outline-danger">
                <span class="fa fa-trash"></span>
              </button>
            </td>
          </tr>
          </tbody>
        </table>
      </div>
      <div class="col-3">
        <button (click)="onToggleCart()" class="btn btn-danger btn-block">
          <span class="fa fa-shopping-basket"></span>
          [{{cartItemsCount}}] Your Cart Items
        </button>
      </div>
    </div>
</div>
```

Task:



localhost:4200

Amazon Shopping

elect Category

Electronics

elect Product

JBL Speaker

review

JBL Speaker

₹4,500.55

Quantity: 2

Add to Cart

Name	Qty	Price	Preview
Nike Casuals		6500.55	
JBL Speaker		4500.55	
Total amount		11,000	

Your Cart Items

[2] Your Cart Items

delete all

Ex: How we can send all details of iterating item?

- You have to pass the iterating object as argument in any specific event.

Likesdemo.component.ts

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

@Component({
  selector: 'app-likesdemo',
  templateUrl: './likesdemo.component.html',
  styleUrls: ['./likesdemo.component.css']
})
export class LikesdemoComponent{
  products = [
    {Name: 'JBL Speaker', Photo: 'assets/speaker.jpg', Likes: 0, Dislikes: 
0},
    {Name: 'Nike Casuals', Photo: 'assets/shoe.jpg', Likes: 0, Dislikes: 
0},
    {Name: 'Shirt', Photo: 'assets/shirt.jpg', Likes: 0, Dislikes: 0},
  ];
  onLikesClick(item){
    item.Likes++;
  }
  onDislikesClick(item){
    item.Dislikes++;
  }
}
```

Likesdemo.component.html

```
<h2>Products Catalog</h2>
<div class="card-deck">
  <div class='card' *ngFor="let item of products">
    <div class="card-header">
      <h3>{{item.Name}}</h3>
    </div>
    <div class="card-body text-center">
      <img [src]="item.Photo" width="200" height="200">
    </div>
    <div class="card-footer text-center">
      <div class="btn-group">
        <button (click)="onLikesClick(item)" class="btn btn-outline-
danger">
          <span class="fa fa-thumbs-up"></span> {{item.Likes}}
Like(s)
        </button>
        <button (click)="onDislikesClick(item)" class="btn btn-
outline-danger">
          <span class="fa fa-thumbs-down"></span> {{item.Dislikes}}
Dislike(s)
        </button>
      </div>
    </div>
  </div>
</div>
```

Ex: Why we need odd, even, first and last properties of “NgFor”?

- You can configure effects dynamically for items based on their occurrence.
- All these values return boolean true or false.

Likesdemo.component.html

```
<div class="form-group">
 <table class="table table-hover">
  <thead>
    <tr>
      <th>Name</th>
      <th>Likes</th>
      <th>Dislikes</th>
      <th>Even</th>
      <th>Odd</th>
      <th>First</th>
      <th>Last</th>
    </tr>
  </thead>
  <tbody>
    <tr [class.oddstyle]="o" [class.evenstyle]="e" *ngFor="let item of
products; let e=even; let o=odd; let f=first; let l=last">
      <td>{{item.Name}}</td>
      <td>{{item.Likes}}</td>
      <td>{{item.Dislikes}}</td>
      <td>{{e}}</td>
```

```
          <td>{{o}}</td>
          <td>{{f}}</td>
          <td>{{l}}</td>
        </tr>
    </tbody>
  </table>
</div>
```

Likesdemo.component.css

```
.oddstyle {
    background-color: rgb(81, 227, 230);
}
.evenstyle {
    background-color: rgb(210, 247, 248);
}
thead > tr {
    background-color: darkcyan;
    color:white;
}
```

Add following extension into visual studio code for CSS intelliSense from extension.

IntelliSense for CSS class names in HTML

Iteration – Track By

- Iteration in UI is controlled by “ngFor”.

- It performs iteration over elements in a collection every time when requested.
- TrackBy identifies the changes in collection.
- TrackBy will notify the changes in collection to iterator.
- Iterator will perform iteration only on the new item and adds to UI.

Trackbydemo.component.ts

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


@Component({
  selector: 'app-trackbydemo',
  templateUrl: './trackbydemo.component.html',
  styleUrls: ['./trackbydemo.component.css']
})
export class TrackbydemoComponent {
  products = [
    {Id: 1, Name: 'TV', Price: 34000.49},
    {Id: 2, Name: 'Mobile', Price: 23000.44}
  ];
  AddNewProduct(){
    this.products = [
      {Id: 1, Name: 'TV', Price: 34000.49},
      {Id: 2, Name: 'Mobile', Price: 23000.44},
      {Id: 3, Name: 'Shoe', Price: 4500.44}
    ];
  }
  TrackChange(index) {
    return index;
  }
}
```

Trackbydemo.component.html

```
<h2>Product Details <button (click)="AddNewProduct()">Add
Product</button></h2>
<table class="table table-hover">
  <thead>
     <tr>
        <th>Product Id</th>
        <th>Name</th>
        <th>Price</th>
     </tr>
  </thead>
  <tbody>
     <tr *ngFor="let item of products; trackBy:TrackChange">
       <td>{{item.Id}}</td>
       <td>{{item.Name}}</td>
       <td>{{item.Price}}</td>
     </tr>
  </tbody>
</table>
```

Iterations and Conditions

- NgFor is for iterations.
- NgIf is for conditions.

- You can handle conditions within iterations.
- You can’t bind both in one element. You should use containers.

Ex:

Conditions.component.ts

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


@Component({
  selector: 'app-conditions',
  templateUrl: './conditions.component.html',
  styleUrls: ['./conditions.component.css']
})
export class ConditionsComponent{
  products = [
    {Name: 'Earpods', Price: 4500.44, Photo: 'assets/earpods.jpg', Category: 'Electronics'},
    {Name: 'JBL Speaker', Price: 6500.44, Photo: 'assets/speaker.jpg', Category: 'Electronics'},
    {Name: 'Nike Casuals', Price: 5500.44, Photo: 'assets/shoe.jpg', Category: 'Footwear'},
    {Name: 'Lee Boot', Price: 2500.44, Photo: 'assets/shoe1.jpg', Category: 'Footwear'},
    {Name: 'Shirt', Price: 1500.44, Photo: 'assets/shirt.jpg', Category: 'Fashion'},
    {Name: 'Jeans', Price: 3500.44, Photo: 'assets/jeans.jpg', Category: 'Fashion'}
  ];
  categories = ['All', 'Electronics', 'Footwear', 'Fashion'];
  selectedCategory = 'All';
}
```

Conditions.component.html

```
<div>
  <h2 class="text-center text-primary">Amazon Shopping</h2>
  <div class="row">
    <div class="col-2">
      <div class="form-group">
       <label>Select a Category</label>
       <div>
         <select [(ngModel)]="selectedCategory" class="form-control">
           <option *ngFor="let item of categories">
             {{item}}
           </option>
         </select>
       </div>
      </div>
      <div class="form-group">
        <label>Select Category</label>
        <div>
          <ul class="list-unstyled">
            <li><input type="radio" value="All" name="opt"
[(ngModel)]="selectedCategory"> All</li>
            <li><input type="radio" value="Electronics" name="opt"
[(ngModel)]="selectedCategory"> Electronics</li>
            <li><input type="radio" value="Footwear" name="opt"
[(ngModel)]="selectedCategory"> Footwear</li>
            <li><input type="radio" value="Fashion" name="opt"
[(ngModel)]="selectedCategory"> Fashion</li>
          </ul>
          </div>
        </div>
      </div>
      <div class="col-10">
         <div class="card-deck">
            <ng-container *ngFor="let item of products">
               <div class="card" *ngIf="selectedCategory=='All' ||
selectedCategory==item.Category || txtSearch==item.Name">
                  <div class="card-header">
                     <h3>{{item.Name}}</h3>
                  </div>
                  <div class="card-body">
                     <img [src]="item.Photo" width="100" height="100" >
                  </div>
                  <div class="card-footer">
                     <h4>{{item.Price}}</h4>
                  </div>
               </div>
            </ng-container>
         </div>
      </div>
   </div>
</div>
```

Attribute Directives

- Attribute directive allows to extend HTML element.
- It makes HTML more declarative.
- It converts the static DOM element into dynamic DOM.

- Angular attribute directives
    - NgModel
    - NgClass
    - NgStyle

NgModel:

- It is an attribute directive that extends HTML element and configures as dynamic element.
- NgModel defines a model reference for HTML element.
- So that it can store the value dynamically and used in UI.

Syntax:

```
<input type="text" [(ngModel)]="username">
```

NgClass:

- It is an attribute directive use to assign a CSS class dynamically to any element.
- It can change the appearance of HTML element dynamically.
- You can apply any CSS class dynamically by using 3 types of references
    - String Reference
    - Array Reference
    - Object Reference

String Reference:

- It allows to define any one CSS class to element dynamically.
  Syntax:
  `<div  [ngClass]=" 'className' " > Your content </div>`

Ex:

Classdemo.component.css

```
.dark {
   background-color:green;
   color: white;
   text-align: center;
   border:2px solid black;
   padding: 10px;
}
.light {
   background-color: lightgreen;
   color: white;
   text-align: center;
   border:2px solid green;
   padding: 10px;
}
```

Classdemo.component.ts

```
export class ClassdemoComponent{
  className = 'effects';
}
```

Classdemo.component.html

```
   <div>
      <h2>Select Theme</h2>
      <div class="form-group">
         <select [(ngModel)]="className" class="form-control">
            <option value="dark">Dark Theme</option>
            <option value="light">Light Theme</option>
         </select>
      </div>
      <h1 [ngClass]="className">Sample Text</h1>
   </div>
```

Array Reference

- It allows to define multiple classes to one element.

Syntax:

```
 <div [ngClass]=”[‘class1’, ‘class2’]”> </div>
```

Ex:

Classdemo.component.css

```
.text-effects {
   text-align: center;
}
.border-effects {
   border:2px solid darkcyan;
}
.shadow-effects {
   box-shadow: 3px 4px 4px darkcyan;
}
```

Classdemo.component.ts

```
export class ClassdemoComponent{
  className = [];
}
```

Classdemo.component.html

```
<div>
        <h2>Type Effects</h2>
        <input [(ngModel)]="className" placeholder="eg: text-effects, border-
effects, shadow-effects" type="text" class="form-control">
        <h1 [ngClass]="className">Sample Text</h1>
 </div>
```

Object Reference

- It is used to turn ON or OFF the effects.

Syntax:

```
<div  [ngClass]=”{‘className’:true, ‘className’:false}”> </div>
```

Ex:

Classdemo.component.ts

```
export class ClassdemoComponent{
  isBorder = false;
  isShadow = false;
  isText = false;
}
```

Classdemo.component.html

```
   <div>
       <h2>Choose Effects</h2>
       <div>
          <ul class="list-unstyled">
             <li><input [(ngModel)]="isBorder" type="checkbox">Border</li>
             <li><input [(ngModel)]="isText" type="checkbox">Text</li>
             <li><input [(ngModel)]="isShadow" type="checkbox">Shadow</li>
          </ul>
       </div>
       <h1 [ngClass]="{'border-effects':isBorder, 'text-effects':isText, 'shadow-
effects': isShadow}">Sample Text</h1>
   </div>
```

NgStyle

- It is used to configure inline styles for HTML element.
- The styles are defined for element individually.
  <div style=”attribute:value”> </div>

- If styles are for specific element and doesn’t require reusability across other elements then you can define inline styles with “NgStyle”

  Syntax:
  public styleObj = { attribute: value }
  <div [ngStyle]=”styleObj”> </div>

Ex:

Styledemo.component.ts

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


@Component({
  selector: 'app-styledemo',
  templateUrl: './styledemo.component.html',
  styleUrls: ['./styledemo.component.css']
})
export class StyledemoComponent{
  styleObject = {
    'position': 'fixed',
    'top': '',
    'left': ''
  };
  onMouseMove(e) {
     this.styleObject = {
       'position': 'fixed',
       'top': e.clientY + 'px',
       'left': e.clientX + 'px'
     };
  }
}
```

Styledemo.component.html

```
<div (mousemove)="onMouseMove($event)" class="container-fluid">
   <div style="height: 1000px;">
   </div>
   <img [ngStyle]="styleObject" src="assets/flag.gif" width="50" height="50">
</div>
```

Ex: Apply Effects Dynamically

Styledemo.component.ts

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

@Component({
  selector: 'app-styledemo',
  templateUrl: './styledemo.component.html',
  styleUrls: ['./styledemo.component.css']
})
```

```
export class StyledemoComponent{
   bgcolorCode = '';
   fgcolorCode = '';
   alignment = 'left';
   styleObj = {
     'background-color': '',
     'color': '',
     'text-align': ''
   };
   ApplyClick(){
     this.styleObj = {
       'background-color': this.bgcolorCode,
       'color': this.fgcolorCode,
       'text-align': this.alignment
     };
   }
}
```

Styledemo.component.html

```
<div class="row">
  <div class="col-3">
    <h2>Choose Effects</h2>
    <div class="form-group">
        <label>Background Color</label>
        <div>
```

```
          <input class="form-control" name="bgcolorCode"
[(ngModel)]="bgcolorCode" type="color">
        </div>
    </div>
    <div class="form-group">
        <label>Foreground Color</label>
        <div>
          <input class="form-control" [(ngModel)]="fgcolorCode"
name="fgcolorCode" type="color">
        </div>
    </div>
    <div class="form-group">
        <label>Alignment</label>
        <select name="alignment" [(ngModel)]="alignment"
class="form-control">
          <option>Left</option>
          <option>Center</option>
          <option>Right</option>
        </select>
    </div>
    <div class="form-group">
        <button (click)="ApplyClick()" class="btn btn-primary btn-
block">Apply Effects</button>
    </div>
  </div>
  <div class="col-9">
    <div style="margin-top: 200px;">
        <h2 [ngStyle]="styleObj">Sample Text</h2>
    </div>
  </div>
</div>
```

Note: NgStyle can also use a method that returns styles.

Ex:

```
ApplyClick(){
     this.styleObj = {
       'background-color': this.bgcolorCode,
       'color': this.fgcolorCode,
       'text-align': this.alignment
     };
     return this.styleObj;
   }
<div style="margin-top: 200px;">
        <h2 [ngStyle]="ApplyClick()">Sample Text</h2>
</div>
```