Component Hierarchy

- Creating components
- Accessing a component in another component
- Transporting data across components
- Defining custom events for components
- Using Material Components
- Component Life Cycle Hooks
- Change Detection
- Content Projection

Access a component in another component

- You can access any component and use in the current component by accessing its selector.
- In Single application user stay on index.html page and can get access to everything onto index page.

Syntax:

```
<app-parent>
   <app-child> </app-child>
</app-parent>
```

Ex:

- **Add following component**

  > ng g c parent

  > ng g c child

child.component.html

```
<div class="container" style="padding: 30px; background-color: green; color:white; text-align: center;">
  <h3>Child Component</h3>
</div>
```

parent.component.html

```
<div class="container-fluid" style="background-color: lightgoldenrodyellow; margin: 20px; height: 200px;">
   <h2>Parent Component</h2>
   <app-child></app-child>
</div>
```

Set parent component in start up

Transporting Data from Parent to Child

- Child component must have property to store the value coming from parent component.
- The properties of one component are not accessible directly to another component, it is local and private to that component.
- If you want any property to access and store value from other component then you have to mark the property by using directive “@Input()”
- @Input() marker will mark the property as global so that it can be access from another component. Input refers to a property that can store data.

- If you are creating a property to receive value then you have to mark it as “@Input()”
- “@Input()” is imported from ‘angular/core’

Ex:

Child.component.ts

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


@Component({
  selector: 'app-child',
  templateUrl: './child.component.html',
  styleUrls: ['./child.component.css']
})
export class ChildComponent {
  @Input() msg = 'child component property';
}
```

Child.component.html

```
<div class="container" style="padding: 30px; background-color: green; color:white; text-align: center;">
  <h3>Child Component</h3>
  <p>{{msg}}</p>
</div>
```

Parent.component.ts

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

```
@Component({
  selector: 'app-parent',
  templateUrl: './parent.component.html',
  styleUrls: ['./parent.component.css']
})
export class ParentComponent {
   helloMsg = 'Hello from Parent';
}
```

Parent.component.html

```
<div class="container-fluid" style="background-color: lightgoldenrodyellow; margin: 20px; height: 200px;">
    <h2>Parent Component</h2>
    <app-child [msg]="helloMsg" ></app-child>
</div>
```

Ex: HTML input and output mechanism

```
<form oninput="x.value=parseInt(a.value)+parseInt(b.value)">
   n1 :
   <input type="text" id="a" name="a" value="0">
   <br>
```

```
    n2 :
    <input type="text" id="b" name="b" value="0">
    <br>
    <output id="x" name="x" for="a+b"> </output>
</form>
```

Transporting Data from child to parent component

- Child component must have a custom event that can emit the value out side the component scope.
- Custom event is configured by implementing “EventEmitter” of ‘@angular/core’
- Custom event can emit value outside the scope by using “@Output()” directive.
- You can emit any type of value, which is access in the parent component by using “$event” [Event Arg]

Ex:

Child.component.ts

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


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

```
})
export class ChildComponent {
  @Input() msg = 'child component property';
  hello = 'Hello from Child';
  @Output() messageToParent: EventEmitter<string> =
new EventEmitter<string>();
  SendValueToParent() {
    this.messageToParent.emit(this.hello);
  }
}
```

Child.component.html

```
<div class="container" style="padding: 30px; background-
color: green; color:white; text-align: center;">
  <h3>Child Component</h3>
  <p>{{msg}}</p>
  <button (click)="SendValueToParent()">Send to
Parent</button>
</div>
```

Parent.component.ts

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

@Component({
  selector: 'app-parent',
```

```
  templateUrl: './parent.component.html',
  styleUrls: ['./parent.component.css']
})
export class ParentComponent {
   helloMsg = 'Hello from Parent';
   msg = '';
   GetfromChild(e){
     this.msg = e;
   }
}
```

Parent.component.html

```
<div class="container-fluid" style="background-color: lightgoldenrodyellow; margin: 20px; height: 400px;">
   <h2>Parent Component</h2>
   <app-child (messageToParent)="GetfromChild($event)" [msg]="helloMsg" ></app-child>
   <p>
      {{msg}}
   </p>
</div>
```

Ex:

- **Add following components**
  > ng g c filter
  > ng g c productslist

Filter.component.ts

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


@Component({
  selector: 'app-filter',
  templateUrl: './filter.component.html',
  styleUrls: ['./filter.component.css']
})
export class FilterComponent{
   @Input() AllCount = 0;
   @Input() ElectronicsCount = 0;
   @Input() FootwearCount = 0;
   @Input() FashionCount = 0;


   categoryName = 'All';


   @Output() SendCategoryName: EventEmitter<string> = new EventEmitter<string>();
```

```
  OnCategoryChange() {
    this.SendCategoryName.emit(this.categoryName);
  }
}
```

Filter.component.html

```
<div>
   <h2>Select Category</h2>
   <select [(ngModel)]="categoryName"
(change)="OnCategoryChange()" class="form-control">
      <option value="All">All [{{AllCount}}]</option>
      <option value="Electronics">Electronics
[{{ElectronicsCount}}]</option>
      <option value="Footwear">Footwear
[{{FootwearCount}}]</option>
      <option value="Fashion">Fashion
[{{FashionCount}}]</option>
   </select>
</div>
```

Productslist.component.ts

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

@Component({
  selector: 'app-productslist',
```

```
  templateUrl: './productslist.component.html',
  styleUrls: ['./productslist.component.css']
})
export class ProductslistComponent {


  products = [
    {Name: 'JBL Speaker', Price: 4500.44, Photo:
'assets/speaker.jpg', Category: 'Electronics'},
    {Name: 'Earpods', Price: 3500.44, Photo:
'assets/earpods.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'},
  ];
  allCount = this.products.length;
  electronicsCount =
this.products.filter(x=>x.Category=='Electronics').length;
  footwearCount =
this.products.filter(x=>x.Category=='Footwear').length;
  fashionCount =
this.products.filter(x=>x.Category=='Fashion').length;
```

```
  CategoryName = 'All';
  OnFilterChange(val) {
    this.CategoryName = val;
  }
}
```

Productslist.component.html

```
<div class="container-fluid">
 <h1 class="text-center text-primary">Amazon Shopping</h1>
 <div class="row">
   <div class="col-3">
     <app-filter (SendCategoryName)="OnFilterChange($event)" [AllCount]="allCount" [ElectronicsCount]="electronicsCount" [FashionCount]="fashionCount" [FootwearCount]="footwearCount"></app-filter>
   </div>
   <div class="col-9">
    <div class="card-deck">
    <ng-container *ngFor="let item of products">
      <div class="card" *ngIf="CategoryName=='All' || CategoryName==item.Category">
          <div class="card-header">
```

```
            <h4>{{item.Name}}</h4>
          </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>
```

Component Life Cycle