11

Component Communication

- **Parent Child**
  - parent to child (@Input)
  - child to parent (@output)
  - Reference (@ViewChild & @ContentChild)
- between the different components using service

1. Binding (@Input & @Output)

- the parent component can pass the data to the child using the @input Property.

```
@Component({
  selector: 'child',
  ...
})
export class ChildComponent {
  @Input() counter: number = 0;
  @Output() counterChange = new EventEmitter<number>();

  onUpdateCounter() {
    this.stateChange.emit(this.counter + 1);
  }
}
```

To bind a value to the input:

```
<child (counterChange)="onCounterChange($event)"></child>
```

To establish a two-way binding:

```
<child [(counter)]="counter"></child>
```

2. Provider (Service)

Provider is a more high-level alternative for communication powered by the Angular **dependency injection**. Instead of establishing a direct linkage between components, a standalone injectable (service) is used as a middleman between them.

```
@Injectable()
export class MiddlemanService {
  isAuthenticated = false
}

@Component(...)
export class LoginComponent {
  constructor(private middlemanService: MiddlemanService) {}

  onLoginSuccess() {
    this.middlemanService.isAuthenticated = true;
  }

  ...
}

@Component({
  template: `
    isAuthenticated: {{ middlemanService.isAuthenticated }}
  `,
  ...
})
export class JustARandomComponent {
  constructor(public middlemanService: MiddlemanService) {}
}
```

3.Reference (@ViewChild & @ContentChild)

In simple, to directly grab the instance reference of the specific component and manipulate it programmatically.

@ViewChild() allows a component to access a child component or DOM element. For example, a parent component might use @ViewChild() to access a child component like this:

In order to use the @ViewChild() decorator, we need to reference the child component in the parent component's class, and then we can use to query the child component's properties and methods or manipulate the DOM directly.