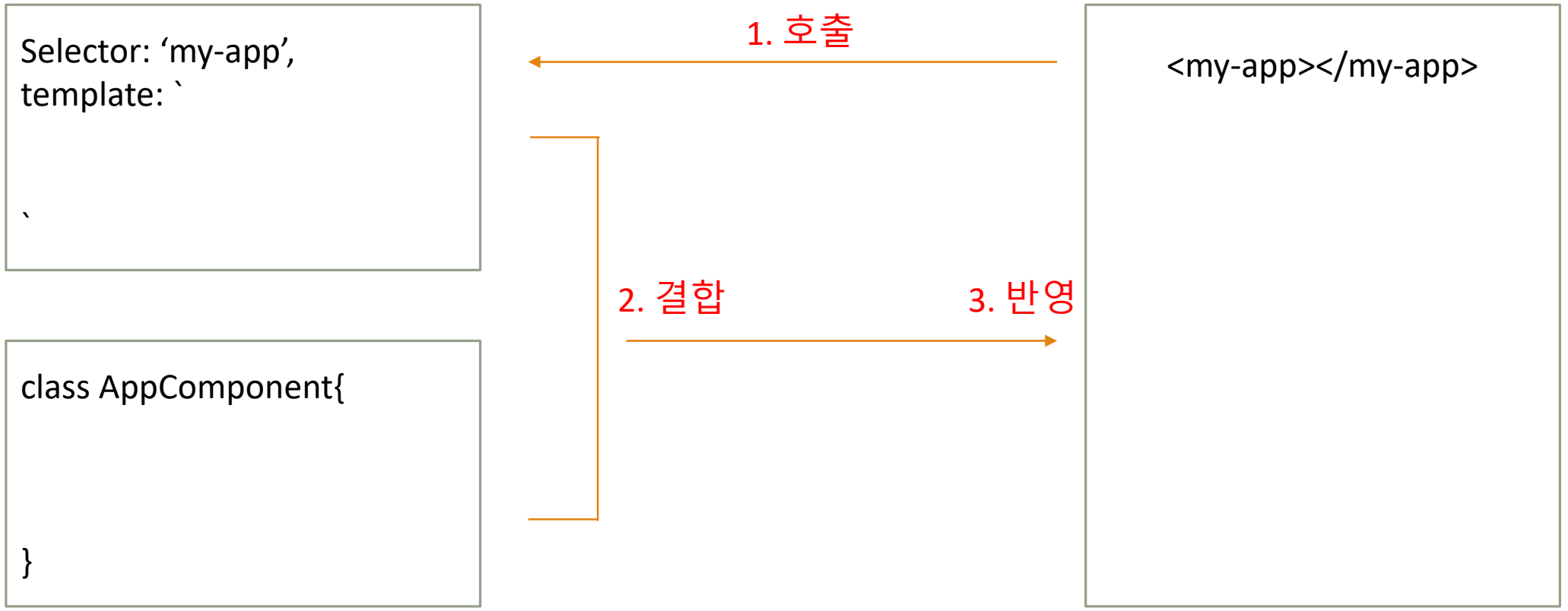
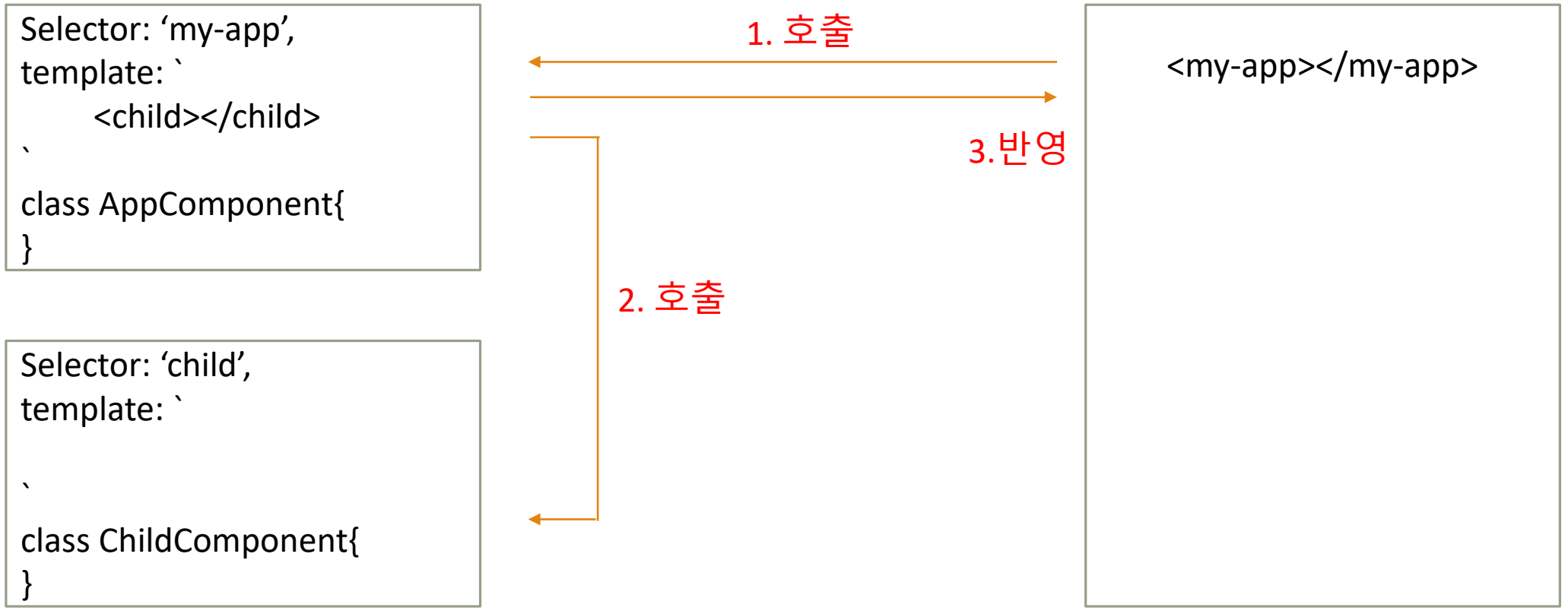
Angular Component

Component


Selector: ‘my-app’,
template: `
`
class AppComponent{
}
1. 호출
2. 결합
3. 반영
<my-app></my-app>

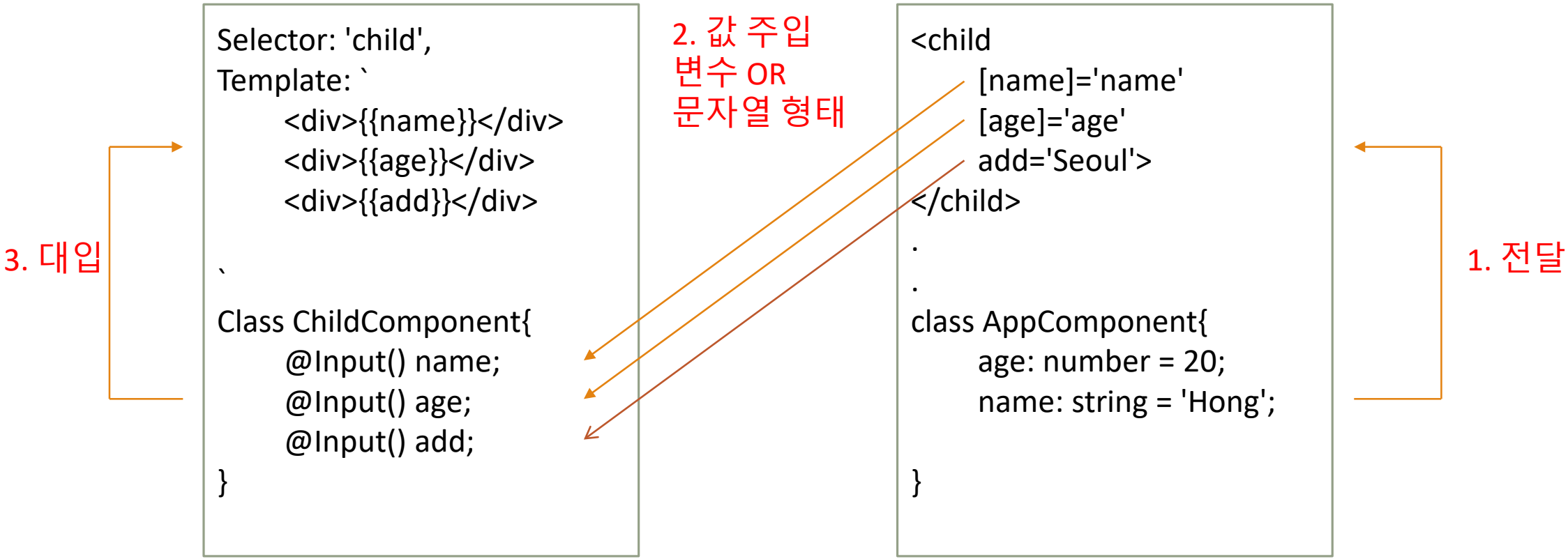
Nested Component



Component 값 전달

1. Component 밖에서 들어오는 데이터는 프로퍼티로 바인딩.
2. 누가 전달하는지는 알 필요가 없으며, 어떤 형식의 데이터인지만 알면 된다.
3. 밖으로 데이터를 전달할 경우는 이벤트를 이용
4. 밖으로 전달할 경우도 어디로 전달되는지 알 필요가 없다. 받는 쪽에서 이벤트를 통해 받음

@Input()


Selector: 'child',
Template: `
{{name}}
{{age}}
{{add}}
`
Class ChildComponent{
@Input() name;
@Input() age;
@Input() add;
}
2. 값 주입
변수 OR
문자열 형태
<child
[name]='name'
[age]='age'
add='Seoul'>
</child>
.
.
class AppComponent{
age: number = 20;
name: string = 'Hong';
}
3. 대입
1. 전달

@Output()

```
Selector: 'child',
Template: `
        <button (click)='sendObj()
'>Click<button>
`
Class ChildComponent{
        @Output() outObj: EventEmitter<any>
                = new EventEmitter<any>();

        sendObj(): void {
                outObj.emit( value );
        }
}
```

1. Event 객체 생성

2. Event 발생

3. Event 전파

```
<child
        (outObj)='getValue($event) '
</child>
.
.
class AppComponent{
        getValue(value: Type) {
                변수 = value
        }

}
```

4. Event Catch 및 처리

@ViewChild

```
Selector: 'child',
Template: `
        …
`

Class ChildComponent{
        public name: string = 'Hong';
        public age: number = 20;
        public user: { id: number, name: string } = {
                id: 10,
                name: 'HongGilDong'
        }
        private num: number = 100;
}
```

public 참조 가능

private 참조 불가

```
Selector: 'parent',
Template: `
        <child #one></child>
`

Class ChildComponent{
        @ViewChild('one') one: A06One;

        ngAfterContentInit() {
                this.name = this.one.name;
                this.age = this.one.age;
                this.user = this.one.user;
        }
}
```

ng-content

```
Selector: 'child',
Template: `
        <ng-content select=".header">

        </ng-content>
        <hr>
        <ng-content select=".footer">

        </ng-content>
`
Class ChildComponent{
}
```

```
Selector: 'parent',
Template: `
        <child>
                <div class="header">
                        Header Content: {{name}}
                </div>
                <div class="footer">
                        Footer Content: {{name}}
                </div>
        </child>
`
Class ChildComponent{
        name: string = 'Parent Component';
```

부모 컴포넌트의 View가 자식 요소에 포함되어 표시 됨