One framework.

Mobile & desktop.

5강. Component 분리

DEVELOP ACROSS ALL PLATFORMS

Learn one way to build applications with Angular and reuse your code and abilities to build apps for any deployment target. For web, mobile web, native mobile and native desktop.

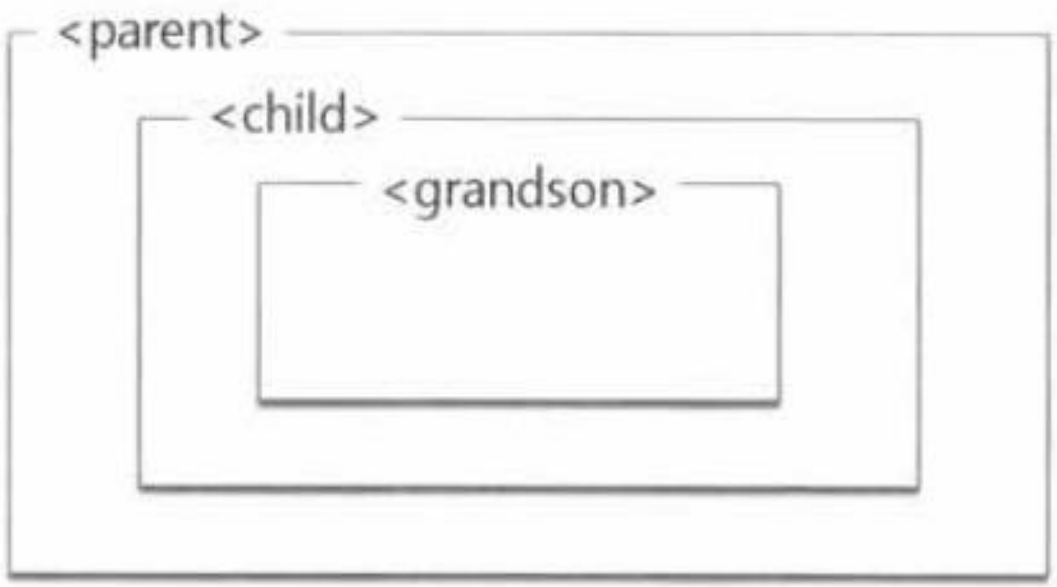
05 강.

Component 분리

❑ 1) Component 분리

* **중첩** 컴포넌트 활용한 레이아웃 작성

컴포넌트는 포함 관계에 따라서 계층구조로 관리됨.



중첩 컴포넌트(부모-자식-손자)

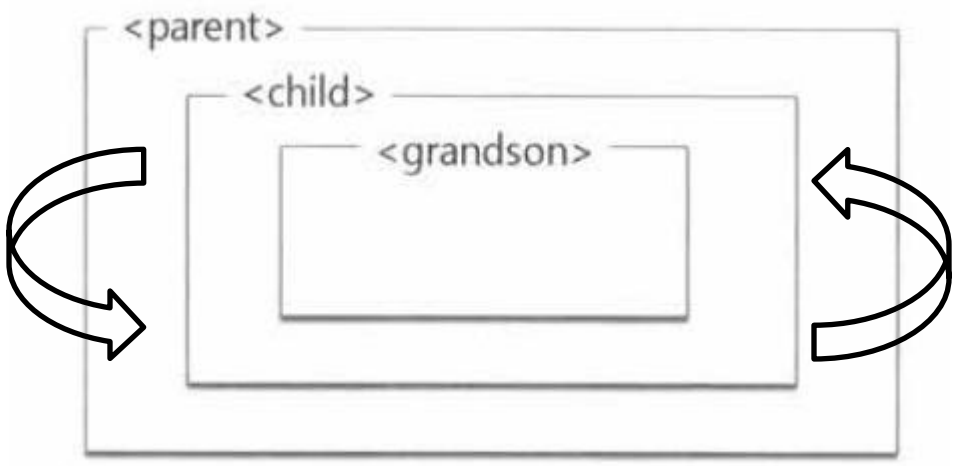
❑ 2) Component 간 데이터 전달 방법

* **중첩** 컴포넌트간의 데이터 전달

부모 컴포넌트--------→ 자식 컴포넌트로 데이터 전송: @Input() +속성

자식 컴포넌트--------→ 부모 컴포넌트로 데이터 전송: @Output() + 이벤트

@Input()
+
속성



중첩 컴포넌트(부모-자식-손자)

@Output()
+
이벤트
(EventEmitter)

❑ 부모->자식 데이터 전달 (child컴포넌트의 구현)

TS child.component.ts ×

src > app > child > TS child.component.ts > ChildComponent

```
1   import { Component, OnInit } from '@angular/core';
2   import {Input} from '@angular/core';
3
4   @Component({
5     selector: 'app-child',
6     templateUrl: './child.component.html',
7     styleUrls: ['./child.component.css']
8   })
9   export class ChildComponent implements OnInit {
10    title="부모 컴포넌트에서 데이터 전달";
11    @Input() username:string;
12    @Input() userage:number;
13
14    constructor() { }
15    ngOnInit(): void {
16    }
17
18  }
```

<> child.component.html × TS child.component.ts

src > app > child > <> child.component.html > br

```
1   <p>{{title}}</p>
2   이름 :{{username}}<br>
3   나이:{{userage}}<br>
4
```

❑ 부모->자식클래스 데이터 전달 , 부모클래스의 구현

child.component.html | app.component.html X | child.component.ts

src > app > app.component.html > app-child

```
1  <h1>부모컴포넌트</h1>
2  <h3>다음은 자식 컴포넌트입니다.</h3>
3  <app-child username="홍길동" userage="20"></app-child>
```

```
export class ChildComponent implements OnInit {
  title="부모 컴포넌트에서 데이터 전달";
  @Input() username:string;
  @Input() userage:number;
```

❑ 3) @Input() + 속성

부모 컴포넌트에서 자식 컴포넌트로 데이터를 전달할 때 사용하는 방법으로
부모는 자식 template의 속성을 이용하여 데이터를 지정하고
자식은 @Input() 을 사용하여 전달된 데이터를 사용한다.


```
export class ChildComponent {

  @Input() username:string;
  @Input() userage:number;

}
```

app.component.html ×

my-app > src > app > app.component.html > ...

```
1  <h1>부모 컴포넌트</h1>
2  <h3>다음은 자식 컴포넌트입니다.</h3>
3  <app-child username="홍길동" userage="20"></app-child>
```

❑ 3) @Input() + 속성 // [전달할 속성명]= "클래스변수명 "

C:\angular_chul\chul-app>ng g component child

```
export class AppComponent {
  title = 'my-app';

  //자식에게 전달할 데이터
  app_username = "홍길동"
  app_userage = 20

}
```

app.component.html ×

y-app > src > app > <> app.component.html > ...

```
1  <h1>부모 컴포넌트</h1>
2  <h3>다음은 자식 컴포넌트입니다.</h3>
3  <app-child [username]="app_username"
4    [userage]="app_userage"></app-child>
```

```
export class ChildComponent {

    @Input() username:string;
    @Input() userage:number;

}
```

child.component.html ×

y-app > src > app > child > <> child.comp

```
1  <p>child works!</p>
2  이름:{{username}}<br>
3  나이:{{userage}}<br>
```

❑ 3) @Input() + 속성

```
export class AppComponent {
  title = 'my-app';

  //자식에게 전달할 데이터
  app_friends=["홍길동","이순신","유관순"];

}
```

app.component.html ×

y-app > src > app > app.component.html > ...

```
1  <h1>부모 컴포넌트</h1>
2  <h3>다음은 자식 컴포넌트입니다.</h3>
3  <app-child [friends]="app_friends"></app-child>
```

```
export class ChildComponent {

    @Input() friends:string[];
}
```

child.component.html ×

y-app > src > app > child > child.component.html > ...

```
1  <p>child works!</p>
2  <div *ngFor="let friend of friends">
3      {{friend}}
4  </div>
```

❑ 실습 문제 1

다음과 같이 중첩된 레이아웃으로 구성된 어플리케이션을 구현 하시오.

- 부모 컴포넌트인 app.component.ts 파일에서 books 데이터를 저장하고 자식 컴포넌트인 book.component.ts에 전달하여 출력한다.

app.component.ts

book.component.ts

모든 데이터는 부모에 저장

```
export class AppComponent {
  title = '도서 목록';

  books =[{id:'p01',name:'위험한 식탁',price:2000,date:'20170401',img:'a.jpg'},
  {id:'p02',name:'공부의 비결',price:3000,date:'20170402',img:'b.jpg'},
  {id:'p03',name:'오메르타',   price:2500,date:'20170401',img:'c.jpg'},
  {id:'p04',name:'행복한 여행',price:4000,date:'20170401',img:'d.jpg'},
  {id:'p05',name:'해커스 토익',price:2000,date:'20170401',img:'e.jpg'},
  {id:'p06',name:'도로 안내서',price:2000,date:'20170401',img:'f.jpg'},
];
```

❑ 실습1

```
C:\angular_chul\chul-app>ng g component book
```

```
import { Component } from '@angular/core';
import {Book} from "./book/Book";
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  titleName:string="도서 목록";
  books:Book[]=[
    {id:"p01", name:"위험한 식탁", price:2000, date:'20180202', img:"a.jpg"},
    {id:"p02", name:"공부의 비결", price:3000, date:'20180201', img:"b.jpg"},
    {id:"p03", name:"오메르타", price:4000, date:'20180203', img:"c.jpg"},
    {id:"p04", name:"행복한 여행", price:5000, date:'20180204', img:"d.jpg"},
    {id:"p05", name:"해커스토익", price:6000, date:'20180205', img:"e.jpg"},
    {id:"p06", name:"여행안내서", price:7000, date:'20180206', img:"f.jpg"},
  ];
}
```

탐색기

열려 있는 편집기

CHUL-APP
- e2e
- node_modules
- src
  - app
    - book
      - book.component.css U
      - book.component.html U
      - book.component.spec.ts U
      - book.component.ts U
      - Book.ts U

TS Book.ts × | TS app.component.

src > app > book > TS Book.ts > Bo

```
export class Book{
    id:string;
    name:string;
    price:number;
    date:string;
    img:string;
}
```

❑ 실습1

app.component.html × | Book.ts | app.component.ts

src > app > app.component.html > app-book

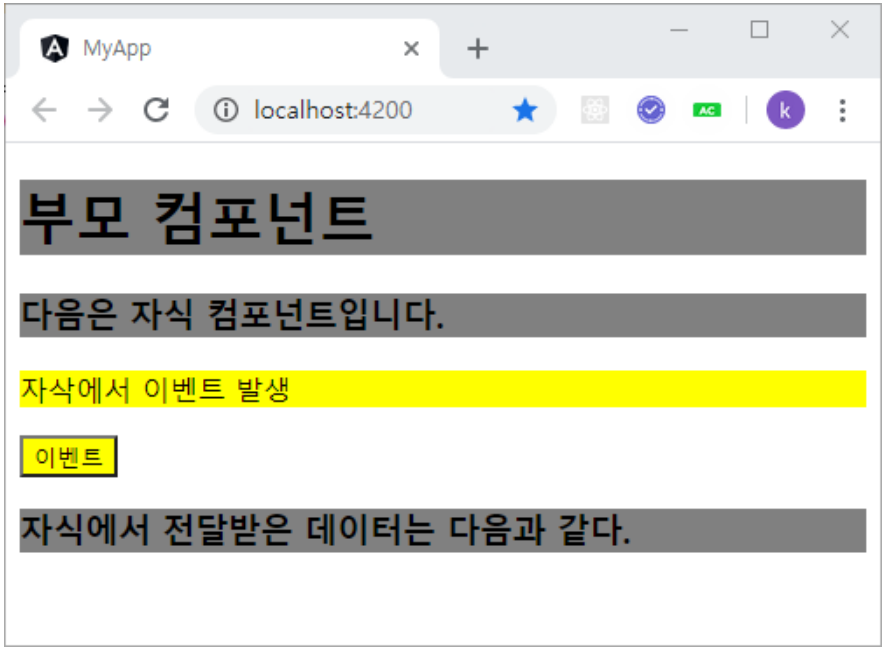
```
<app-book [booklist]="books" [title]="titleName"></app-book>
```

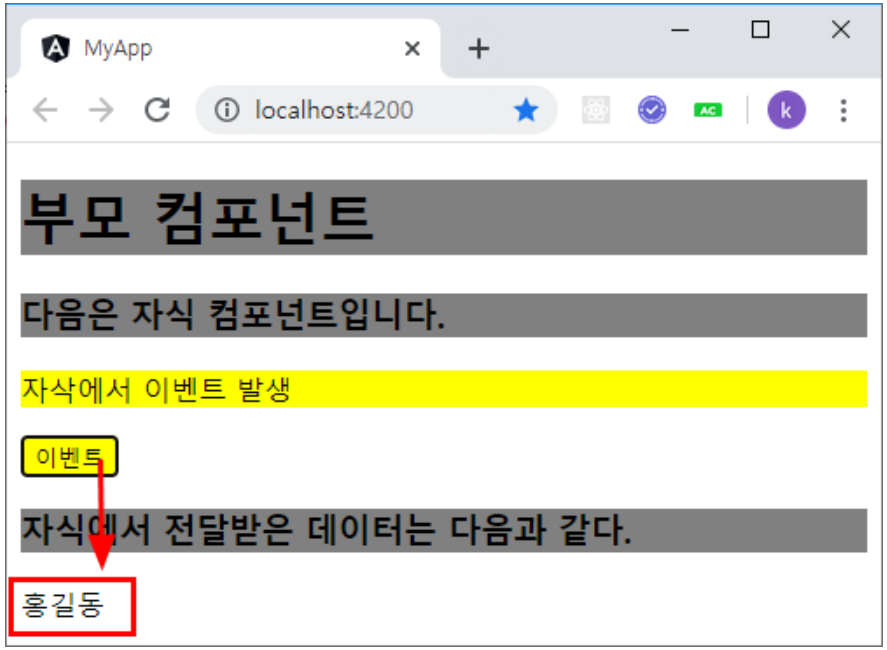
```
import { Book } from './Book';

@Component({
  selector: 'app-book',
  templateUrl: './book.component.html',
  styleUrls: ['./book.component.css']
})
export class BookComponent {
@Input() booklist:Book[];
@Input("title") titleName:string;

}
```

```
<h1>{{titleName}}{{booklist.length}}</h1>
<ul>
    <li *ngFor="let book of booklist">
        <img src="../../assets/image/{{book.img}}" width="100" height="100">{{book.name}}
    </li>
</ul>
```

❑ 4) @Output + 이벤트

자식 컴포넌트에서 부모 컴포넌트로 데이터를 전달할 때 사용하는 방법으로
자식은 사용자 정의 이벤트와 @Output을 사용하여 부모에게 데이터를 전달하고
부모는 template의 속성에 지정된 이벤트를 사용하여 데이터를 이용할 수 있다.





❑ 자식에서 부모에게 데이터 전달 @Output() + EventEmitter.emit(data)

TS child.component.ts ✕ | child.component.html

src > app > child > TS child.component.ts > ChildComponent > constructor

```
1  import { Component, OnInit } from '@angular/core';
2  import {Output, EventEmitter} from '@angular/core';
3
4  @Component({
5    selector: 'app-child',
6    templateUrl: './child.component.html',
7    styleUrls: ['./child.component.css']
8  })
9  export class ChildComponent implements OnInit {
10   //부모에게 사용자 정의 이벤트 전라할 EventEmitter객체
11   @Output() customEvent =    new EventEmitter();
12
13   send(name){
14       this.customEvent.emit(name); //부모에게 데이터 전달
15   }
```

❑ 4) @Output + 이벤트

```
C:\angular_chul\chul-app>ng g component child
```

child.component.ts

src > app > child > child.component.ts > ChildComponent

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

child.component.html

y-app > src > app > child > child.component.html > ...

```
<p>자식에서 이벤트 발생</p>
<button (click)="send('홍길동')">
    이벤트</button>
```

```
export class ChildComponent {

  //부모에게 사용자 정의 이벤트 전달할 EventEmitter 객체
  @Output() customEvent =
            new EventEmitter<string>()

  send(name){
    this.customEvent.emit(name);
  }
}
```

app.component.html

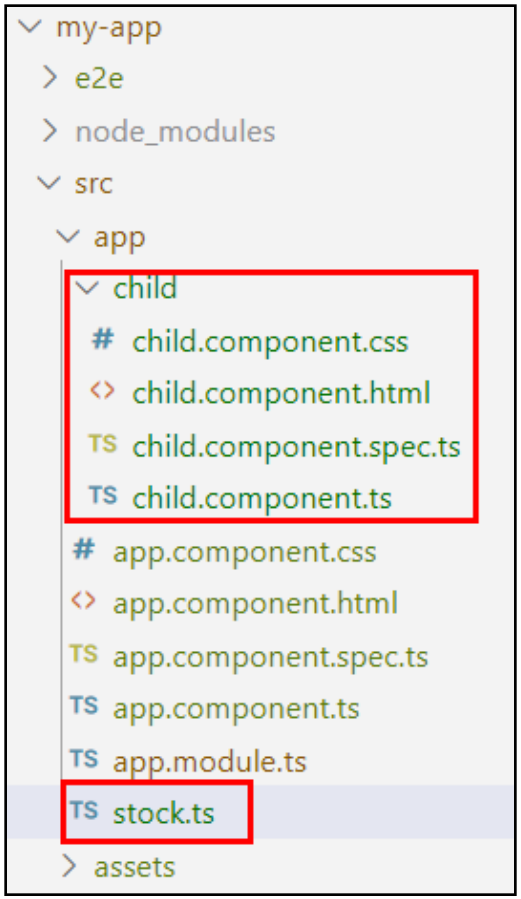
y-app > src > app > app.component.html > ...

```
<h1>부모 컴포넌트</h1>
<h3>다음은 자식 컴포넌트입니다.</h3>
<app-child (customEvent)="handleEvent($event)">
    </app-child>
<h3>자식에서 전달받은 데이터는 다음과 같다.</h3>
{{response}}
```

```
export class AppComponent {
  title = 'my-app';
  response = "";
  handleEvent($event){
    this.response = $event;
  }
}
```

❑ 5) interface 이용한 데이터 전달

자식 컴포넌트와 부모 컴포넌트간의 데이터 전달시, 전달되는 데이터에 대한 규칙이 필요하다. 대표적으로 변수명, 데이터타입등으로서 이것을 강제하기 위하여 인터페이스를 사용한다.




```
C:\angular\my-app>ng g component child
CREATE src/app/child/child.component.html (20 bytes)
CREATE src/app/child/child.component.spec.ts (621 bytes)
CREATE src/app/child/child.component.ts (271 bytes)
CREATE src/app/child/child.component.css (0 bytes)
UPDATE src/app/app.module.ts (392 bytes)

C:\angular\my-app>ng g interface Stock
CREATE src/app/stock.ts (27 bytes)
```

stock.ts

-app > src > app > TS stock.ts > ...

```
1  export interface Stock {
2      stockSymbol:string,
3      stockPrice:number
4  }
```

❑ 5) interface 이용한 데이터 전달

app.component.ts ×

y-app > src > app > TS app.component.ts > ...

```
1   import { Component } from '@angular/core';
2   import {Stock} from'./stock';
3
4   @Component({
5     selector: 'app-root',
6     templateUrl: './app.component.html',
7     styleUrls: ['./app.component.css']
8   })
9   export class AppComponent {
10    title = '인터페이스 이용한 데이터 전달';
11
12    myStock:Stock={
13      stockSymbol:"Angular",
14      stockPrice:100
15    };
16  }
```

app.component.html ×

y-app > src > app > app.component.html > ...

```
1   <h1>{{title}}</h1>
2   <app-child  [iStock]="myStock"></app-child>
```

child.component.ts ×

y-app > src > app > child > TS child.component.ts > ...

```
1   import { Component, OnInit, Input } from '@angul
2   import {Stock} from'../stock';
3
4   @Component({
5     selector: 'app-child',
6     templateUrl: './child.component.html',
7     styleUrls: ['./child.component.css']
8   })
9   export class ChildComponent{
10
11    @Input() iStock:Stock;
12
13    stockSymbol = "";
14    stockPrice = 0;
15    handleEvent(){
16      this.stockSymbol = this.iStock.stockSymbol;
17      this.stockPrice = this.iStock.stockPrice;
18    }
19  }
```

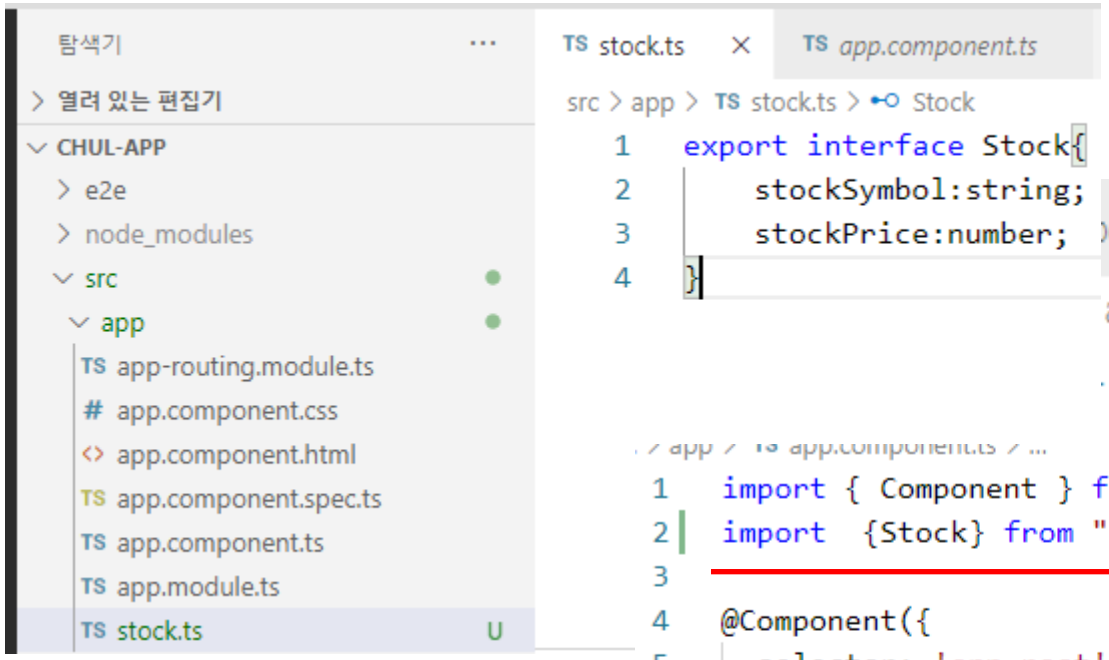
child.component.html ×

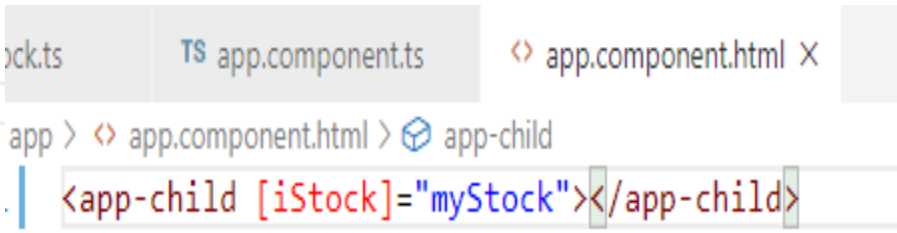
y-app > src > app > child > child.component.html > ...

```
1   <p>child works!</p>
2   <button (click)="handleEvent()">Stock</button><br>
3   주식명:{{stockSymbol}}<br>
4   주식가격:{{stockPrice}}
```

❑ Interface데이터 전달

-stock.ts의 생성






```
import { Component } from '@angular/core';
import  {Stock} from "./stock"

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'chul-app';
  myStock:Stock={
    stockSymbol:"Angular",
    stockPrice:100
  }
```

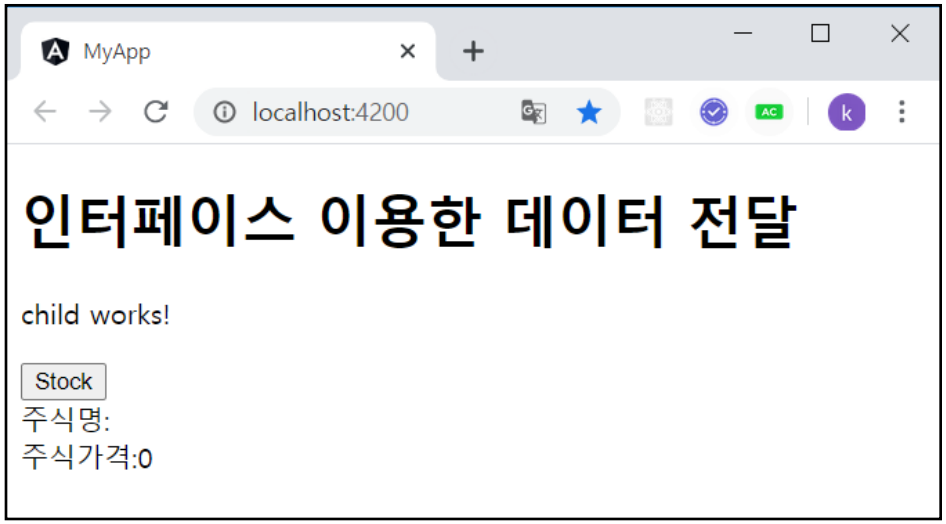
❑ Interface데이터 전달

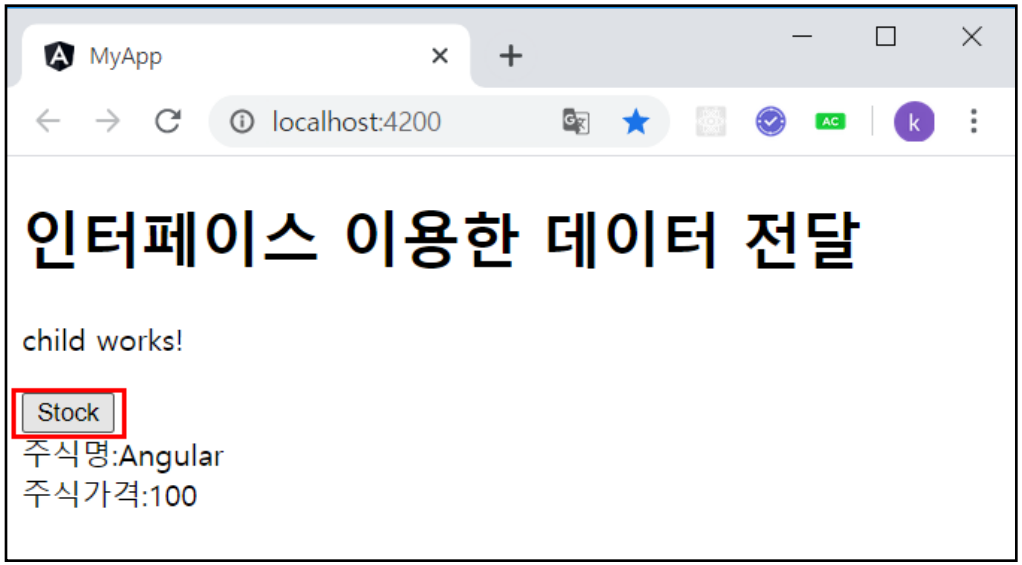
```
C:\angular_chul\chul-app>ng g component child
CREATE src/app/child/child.component.html (20 bytes)
```

```
export class ChildComponent {

  @Input() iStock:Stock;

  stockSymbol="";
  stockPrice=0;

  handleEvent(){
    this.stockSymbol = this.iStock.stockSymbol;
    this.stockPrice = this.iStock.stockPrice;

```

```
<p>child works!</p>
<button (click)="handleEvent()">Stock</button><br>
주식명:{{stockSymbol}}<br>
주식가격:{{stockPrice}}

```

❑ 5) interface 이용한 데이터 전달

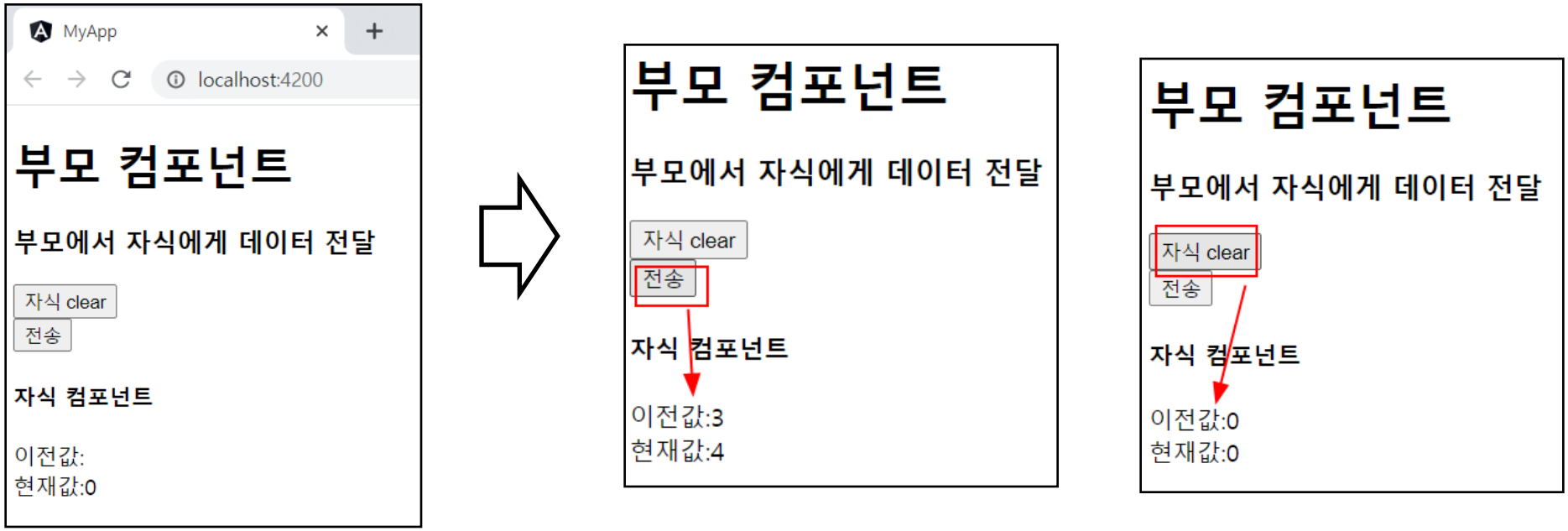




❑ 6) ngOnChanges() 메서드 와 @ViewChild

-부모 컴포넌트에서 자식 요소에 데이터를 전달할 때 호출되는 콜백 메서드이다.
-전달되는 데이터의 이전값과 현재값을 확인할 수 있는 SimpleChanges 객체가 제공된다.
- @ViewChild는 부모 컴포넌트에서 자식요소를 직접 접근할 수 있는 decorator이다.

```
C:\angular_chul\chul-app>ng g component child
```



❑ 자식컴포넌트의 작성

TS child.component.ts × | app.component.html ● | TS app.component.ts | child.component.html

src > app > child > TS child.component.ts > ChildComponent > cur_num

```
1   import { Component, OnInit,Input, SimpleChange } from '@angular/core';
2   //import SimpleChange, Input
3   @Component({
4     selector: 'app-child',
5     templateUrl: './child.component.html',
6     styleUrls: ['./child.component.css']
7   })
8   export class ChildComponent implements OnInit {
9     @Input() send_num:number;  //부모에서 전송되는 데이터
10    prev_num=0;
11    cur_num=0;
12    ngOnChanges(changes:SimpleChange){
13      this.prev_num= changes["send_num"].previousValue;
14      this.cur_num= changes["send_num"].currentValue;
15      console.log(this.prev_num, "\t", this.cur_num);
16    }
```

❑ 부모 컴포넌트의 작성

TS child.component.ts | TS app.component.ts × | app.component.html | child.co

src > app > TS app.component.ts > AppComponent > child

```
1  import { Component, ViewChild } from '@angular/core';
2  //ViewChild import
3  import { ChildComponent } from './child/child.component';
4  //
5  @Component({
6    selector: 'app-root',
7    templateUrl: './app.component.html',
8    styleUrls: ['./app.component.css']
9  })
10 export class AppComponent {
11   title = 'chul-app';
12   num=0;
13   @ViewChild("kkk") child:ChildComponent;
14   clear(){
15     this.child.prev_num=0;
16     this.child.cur_num=0;
17   }
18   send(){
19     this.num++
20   }
```

app.component.html × | TS child.component.ts | TS app.component.ts | ch

src > app > app.component.html > ...

```
1 <h1>부모컴포넌트</h1>
2 <h3>부모컴포넌트에서 자식에게 데이터 전달</h3>
3 <button (click)="clear()">자식 clear</button><br>
4 <button (click)="send()">전송</button><br>
5 <app-child #kkk [send_num]="num"></app-child>
6 <!--@ViewChild("kkk") child:ChildComponent;  임-->
```

❑ 6) ngOnChanges() 메서드 와 @ViewChild

1) 부모 컴포넌트 template를 작성한다.

app.component.html ×

y-app > src > app > app.component.html > button

```
<h1>부모 컴포넌트</h1>
<h3>부모에서 자식에게 데이터 전달</h3>
<button  (click)="clear()">자식 clear</button><br>
<button  (click)="send()">전송</button><br>
<app-child #kkk [send_num]="num"></app-child>
```

2) 부모 컴포넌트를 작성한다.

```
export class AppComponent{

  title = 'my-app';
  num = 0;

  @ViewChild("kkk") child:ChildComponent;
```

```
  clear(){
    this.child.prev_num = 0;
    this.child.cur_num = 0;
  }
  send(){
    this.num++;
  }
}
```

❑ 6) ngOnChanges() 메서드 와 @ViewChild

3) 자식 컴포넌트를 작성한다.

```
export class ChildComponent  {

  @Input() send_num:number;

  prev_num = 0;
  cur_num =0;
  ngOnChanges(changes: SimpleChanges){
      this.prev_num = changes["send_num"].previousValue;
      this.cur_num = changes["send_num"].currentValue;
  }
}
```

4) 자식 컴포넌트 template를 작성한다.

child.component.html ×

y-app > src > app > child > <> child.component.html >

```
1   <h4>자식 컴포넌트</h4>
2   이전값:{{prev_num}}<br>
3   현재값:{{cur_num}}<br>
```

❑ 실습 문제 2

다음 화면에서 특정 도서 이미지를 선택했을 때, **선택된 도서명을 보여주는 어플리케이션을 구현 하시오.**

-선택된 도서명을 보여주는 input태그는 부모 컴포넌트인 app.component.html 에서 설정하고 [실습문제1]을 활용하여 작성한다.

❑ 실습2

```
> book > <> book.component.html > ul > li > img
<h1>{{titleName}}{{booklist.length}}</h1>
<ul>
    <li *ngFor="let book of booklist">
        <img src="../../assets/image/{{book.img}}" width="100" height="100"
        (click)="send(book.name)">{{book.name}}
    </li>
</ul>
```

```
9   export class BookComponent  {
10    //부모-->자식
11  @Input() booklist:Book[];
12  @Input("title") titleName:string;
13  //자식-->부모
14  @Output() customEvent= new EventEmitter<string>();
15  send(name){
16    this.customEvent.emit(name);//name전달
17  }
18
```

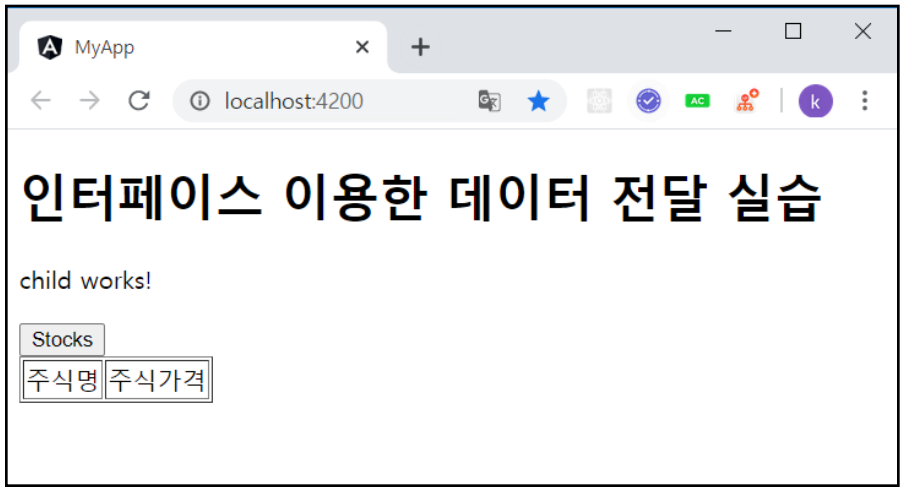
```
selected_name="";
// handleEvent($event){
//   this.selected_name= $event;
// }
handleEvent(name){
  this.selected_name= name;
}
}
```

```
> app > <> app.component.html > ...
1  선택된 도서 : <input type="text" value={{selected_name}} ><br>
2  <app-book [booklist]="books" [title]="titleName" (customEvent)="handleEvent($event)"></app-book>
3
```

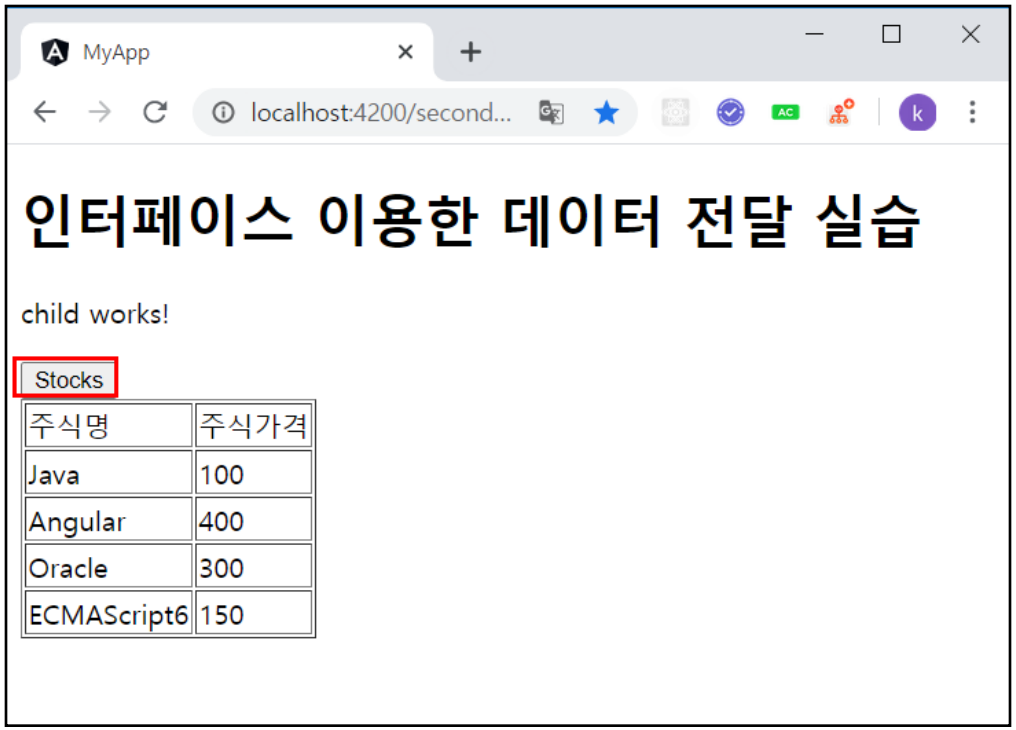
❑ 실습 문제 3

다음과 같이 Stocks 버튼을 선택 했을 때, **주식정보를 보여주는 어플리케이션을 구현 하시오. 이때 주식 정보는 다음 코드를 활용한다.**

```
myStocks:Stock[]=[{stockSymbol:'Java', stockPrice:100},
{stockSymbol:'Angular', stockPrice:400},
{stockSymbol:'Oracle', stockPrice:300},
{stockSymbol:'ECMAScript6', stockPrice:150}]
```




```
PS C:\Angular_Study_chul_2\chul-app> ng g interface stock
CREATE src/app/stock.ts (27 bytes)
PS C:\Angular_Study_chul_2\chul-app>
```



❑ 실습3

```
C:\angular_chul\chul-app>ng g component child
CREATE src/app/child/child.component.html (20 bytes)
```

```
import {Stock}  from "./child/stock"
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'chul-app';

  myStock:Stock[]=[
    {stockSymbol:"Angular", stockPrice:100},
    {stockSymbol:"Oracle", stockPrice:200},
    {stockSymbol:"ReactJs", stockPrice:300},
    {stockSymbol:"Vue", stockPrice:400}
  ];
```

.ts | TS app.component.ts | <> app.component.html ×

op > <> app.component.html > app-child

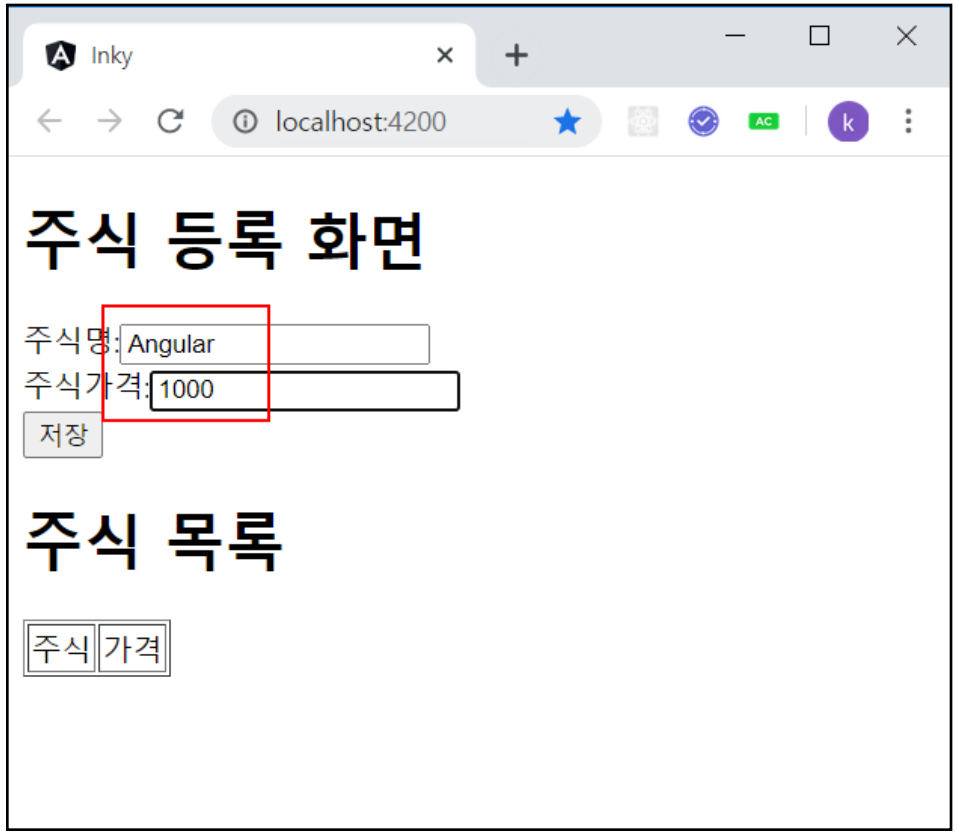
```
<app-child [iStocks]="myStock"></app-child>
```

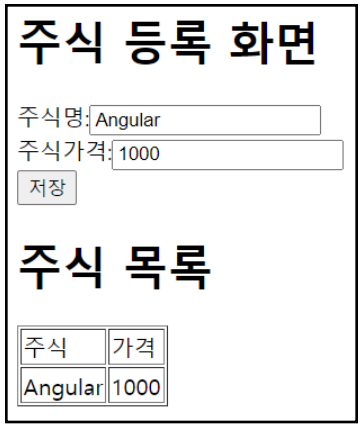
```
@Component({
  selector: 'app-child',
  templateUrl: './child.component.html',
  styleUrls: ['./child.component.css']
})
export class ChildComponent  {
  @Input() iStocks:Stock[];
  showStocks:Stock[];
  handleEvent(){
    this.showStocks= this.iStocks;
  }
```

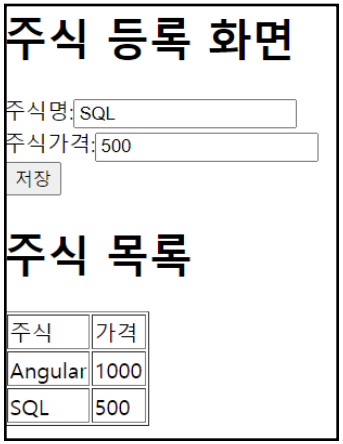
```
<p>child works!</p>
<button (click)="handleEvent()">stock</button>
<table border="1">
    <tr>
        <th>주식명</th>
        <th>주식가격</th>
    </tr>
    <tr *ngFor="let stock of showStocks">
        <td>{{stock.stockSymbol}}</td>
        <td>{{stock.stockPrice}}</td>
    </tr>
</table>
```

❑ 실습 문제 4

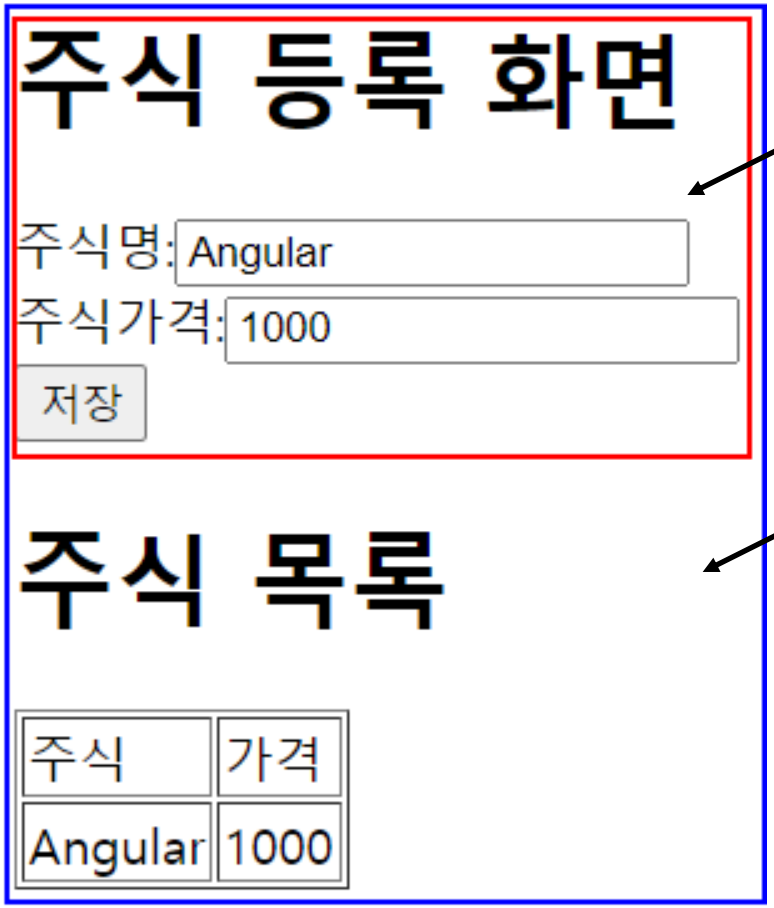
다음 화면과 같이 주식정보를 입력하고 저장버튼을 선택하면 주식 목록에 추가되어 출력되는 어플리케이션을 구현 하시오. (유효성 검증은 구현에서 제외)







❑ 실습 문제 4



StockAddComponent

AppComponent

stock.ts

```
export class Stock {
    stockSymbol:string;
    stockPrice:number;
}
```

❑ 실습4

```
C:\angular_chul\chul-app>ng g component stock-add
CREATE src/app/stock-add/stock-add.component.html (24 byt
```

(F) 편집(E) 선택 영역(S) 보기(V) 이동(G) 실행(R) 터미널(T) 도움말(H)

탐색기 ···

열려 있는 편집기

CHUL-APP
- e2e
- node_modules
- src
  - app
    - stock-add
    - app-routing.module.ts
    - app.component.css
    - app.component.html
    - app.component.spec.ts
    - app.component.ts
    - app.module.ts
    - stock.ts

TS stock.ts ×

src > app > TS stock.ts > Stock > stock

```ts
export class Stock{
    stockSymbol:string;
    stockPrice:number;
}
```

- stock-add .html – stock 객체로 data전송

```html
<h1>주식 등록 화면</h1>
<div>
    주식명:<input type="text" #name><br>
    주식가격:<input type="text" #price><br>
    <button (click)="add({'stockSymbol':name.value, 'stockPrice':price.value})">저장</button>
</div>
```

❑ 실습4

```
export class StockAddComponent  {

    stock:Stock;

    // 1.stock에 데이터 저장
    //2. 부모컴포넌트에게 stock 전달. => @Output + 이벤트(EventEmitter)
   @Output() customEvent = new EventEmitter<Stock>(); //  <부모에게전달할 객체타입>

    add(s:Stock){
      this.stock = s;
      this.customEvent.emit(this.stock); // emit(단하나의값)
    }

}
```

```
<app-stock-add (customEvent)="handleEvent($event)"></app-stock-add>
<h1>주식 목록</h1>
<table border="1">
  <tr>
    <th>주식명</th>
    <th>주식가격</th>
  </tr>
  <tr *ngFor="let stock of stocks">
    <td>{{stock.stockSymbol}}</td>
    <td>{{stock.stockPrice}}</td>
  </tr>
</table>
```

❑ 실습4

```
2   import { Stock } from './stock';
3
4
5
6   @Component({
7     selector: 'app-root',
8     templateUrl: './app.component.html',
9     styleUrls: ['./app.component.css']
0   })
1   export class AppComponent {
2     title = 'my-app2';
3
4     //Stock배열 ==> stock배열을 app.component.html에서 바인딩해서 출력.
5     stocks:Stock[]=[];
6
7     handleEvent(stock:Stock){
8       this.stocks.push(stock);
9     }
0   }
1
```

수고하셨습니다.