



**One framework.
Mobile & desktop.**

4강. 바인딩



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.

04 강.

바인딩 (binding)

□ 1) 바인딩 (binding) 개요

* 바인딩 개요

컴포넌트는 화면 구현을 담당하는 template과 비즈니스로직을 담당하는 컴포넌트 클래스 간의 상호 작용을 위한 방법으로 바인딩(binding)을 이용한다.

종류	설명
{{ }}	단방향, 인터폴레이션이라고 부름. 클래스 속성 및 메서드를 <u>template</u> 에서 접근 가능.
[속성]	속성 바인딩, 태그안에서 사용되고 속성값으로 컴포넌트 클래스의 속성값과 연동
(이벤트)	이벤트 바인딩, 태그안에서 사용되고 template에서 발생하는 이벤트를 컴포넌트 클래스의 메서드와 연동된다.
[class.XX]	class 바인딩
[style.xx]	스타일 바인딩
[(ngModel)]	양방향 바인딩

□ 2) 인터폴레이션 : {{}}

가. {{}}

컴포넌트 클래스의 데이터를
template의 html에게 전달할 때 사용된다. 메소드도 호출 가능함

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

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

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

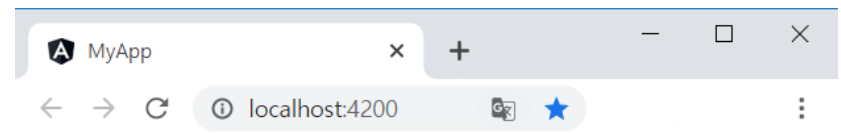
□ 2) 인터폴레이션 : {{ }}

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

```
export class AppComponent {
  title = '바인딩 실습';
  name = "홍길동";
  age = 20;
  address = "서울";
}
```

```
PS C:\Angular_Study_chul_2\chul-app> ng serve
```

```
app.component.html >
y-app > src > app > <> app.component.html > ...
1 <h1>{{title}}</h1>
2 <ul>
3   <li>이름:{{name}}</li>
4   <li>나이:{{age}}</li>
5   <li>주소:{{address}}</li>
6 </ul>
```



바인딩 실습

- 이름:홍길동
- 나이:20
- 주소:서울

□ 2) 인터폴레이션 : Person객체의 사용{}

-person.ts의 작성

```
export class Person{  
    // public name:string;  
    // public age:number;  
    // public address:string;  
    // constructor(name:string, age:number, address:string){  
    //     this.name= name;  
    //     this.age= age;  
    //     this.address= address;  
    // }  
    constructor(public name:string, public age:number, public address:string){}
```

App-component.ts 에서 import후 사용

```
2 | import {Person} from "../person";  
3 |  
4 | @Component({  
5 |     selector: 'app-root',  
6 |     templateUrl: './app.component.html',  
7 |     styleUrls: ['./app.component.css']  
8 | })  
9 | export class AppComponent {  
0 |     title = 'chul-app';  
1 |  
2 |     user:Person={  
3 |         name:"이순신",  
4 |         age:20,  
5 |         address:"서울";  
6 |     }  
7 | }
```

□ 인터폴레이션 : Person객체의 사용{{}}

```

1 <h1>인터폴레이션의 실습</h1>
2 이름{{user.name}}<br>
3 나이{{user.age}}<br>
4 주소{{user.address}}<br>

```

인터폴레이션의 실습

이름이순신
나이20
주소서울

□ 2) 인터플레이션 : {{}}



```
export class BookComponent{

  titleName = "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'},
          ];

  getTitleName(){
    return this.titleName;
  }
}
```

```
<h1>{{getTitleName()}} {{books.length}}권</h1>
<ul>
  <li *ngFor="let book of books">
    
    {{book.name}}
  </li>
</ul>
```




```
C:\angular_chul\chul-app>ng g component book
CREATE src/app/book/book.component.html (19 bytes)
```

```
7  })
8  export class BookComponent {
9      titleName:string="도서목록";
10
11      books=[ //Json의 배열
12          {id:"p01", name:"위험한 식탁", price:2000, date:"20170202", img:"a.jpg"},
13          {id:"p02", name:"공부의 비결", price:3000, date:"20170202", img:"b.jpg"},
14          {id:"p03", name:"오메르타", price:2000, date:"20200202", img:"c.jpg"},
15          {id:"p04", name:"행복한 여행", price:2000, date:"20180202", img:"d.jpg"},
16          {id:"p05", name:"해커스 토익", price:2000, date:"20200202", img:"e.jpg"},
17          {id:"p06", name:"도로안내서", price:2000, date:"20150202", img:"f.jpg"},
18      ];
19      getTitleName():string{
20          return this.titleName;
21      }
22  }
```

Book.component.html의 수정

```

app.component.ts  index.html  app.component.html  TS book.component.ts  book.component.html X
app > book > <> book.component.html > ul
1  <p>{{getTitleName()}},{{books.length}}권</p>
2  <ul>
3      <li *ngFor="let book of books">
4          
5          {{books.name}}
6      </li>
7  </ul>
3

```

App.modules.ts

```

4  import { AppRoutingModule } from './app-routing.module';
5  import { AppComponent } from './app.component';
6  import { BookComponent } from './book/book.component';
7
8  @NgModule({
9      declarations: [
10         AppComponent,
11         BookComponent
12     ],
13     imports: [
14         BrowserModule,
15         AppRoutingModule
16     ]

```

app.component.html의 수정

```

app.component.ts  index.html  app.component.html X
src > app > <> app.component.html > ...
1  <app-book></app-book>
2

```

Main.ts

```

5
7  if (environment.production) {
3    enableProdMode();
3  }
3
1  platformBrowserDynamic().bootstrapModule(AppModule)
2    .catch(err => console.error(err));
3

```

⌘

C:\angular_chul\chul-app>ng serve --open

실행(R) 터미널(T) 도움말(H) index.html

app.component.ts <> app.component.html <> index.html X TS

> <> index.html > ...

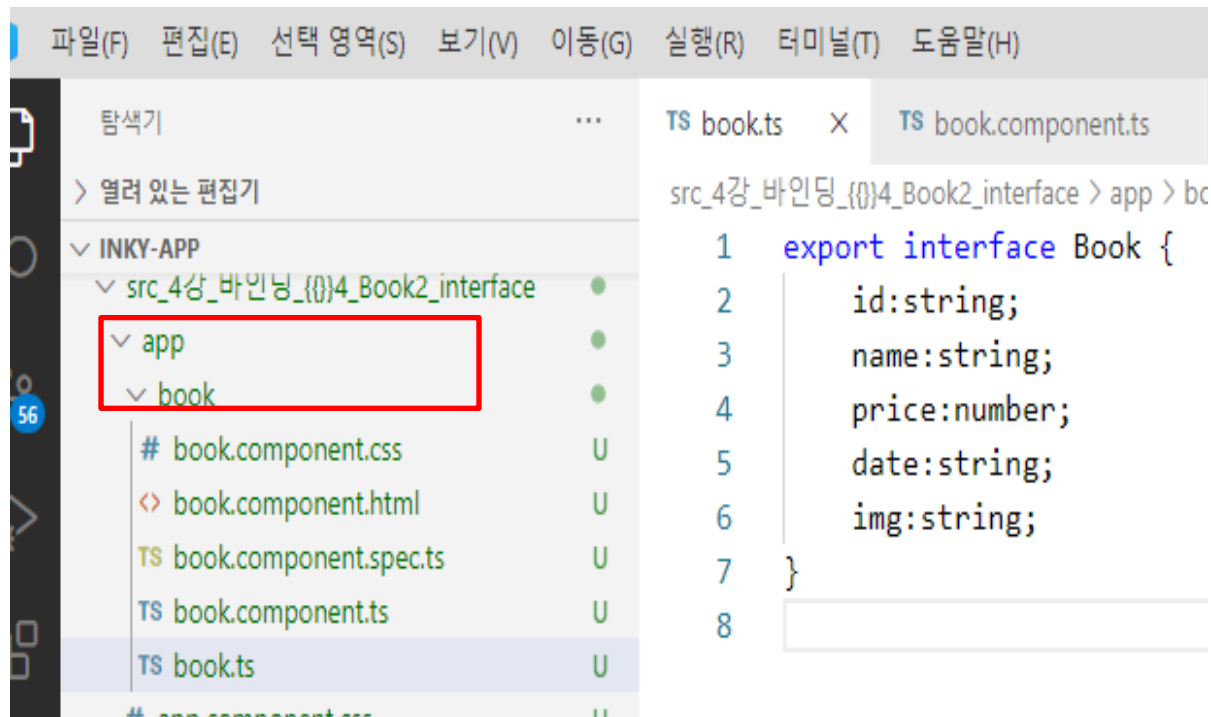
```

4    <meta charset="utf-8">
5    <title>ChulApp</title>
6    <base href="/">
7    <meta name="viewport" content="width=device-w
8    <link rel="icon" type="image/x-icon" href="fa
9  </head>
10 <body>
11   <app-root></app-root>
12 </body>
13 </html>
14

```

□ 실습- *interface* 선언과 사용

```
PS C:\Angular_Study_chul_2\chul-app\src\app> cd book
PS C:\Angular_Study_chul_2\chul-app\src\app\book> ng generate interface book
```





□ Interface의 사용

Interface import , 객체 만들기

```
1 import { Component, OnInit } from '@angular/core';
2 import { Book } from "../book";
3
4 @Component({
5   selector: 'app-book',
6   templateUrl: './book.component.html',
7   styleUrls: ['./book.component.css']
8 })
9 export class BookComponent {
10   titleName:string="도서목록";
11
12   //인터페이스를 이용해서 특정 key값만 사용하도록 강제
13   books:Book[]=[
14     {id:"p01", name:"위험한 식탁", price:2000, date:'20170202', img:'a.jpg'},
15     {id:"p02", name:"공부의 비결", price:300, date:'20170202', img:'b.jpg'},
16     {id:"p03", name:"오메르타", price:2400, date:'20201202', img:'c.jpg'},
17     {id:"p04", name:"행복한 여행", price:5000, date:'20180202', img:'d.jpg'},
18     {id:"p05", name:"해커스토익", price:1000, date:'20130202', img:'e.jpg'},
19     {id:"p06", name:"도로안내서", price:2000, date:'20150202', img:'f.jpg'},
20   ];
21
22   getTitleName(): string{
23     return this.titleName;
24   }
25 }
```

□ *Interface*의 사용



```
src > app > book > <> book.component.html > ul > li
```

```
1 <h1>{{getTitleName()}}{{books.length}}</h1>
2 <ul>
3   <li *ngFor="let book of books">
4     
5     {{book.name}}
6   </li>
7 </ul>
8
```

Ng serve --open

```
book.component.ts <> book.component.html <> app.component.html X
```

```
app > <> app.component.html > app-book
```

```
<app-book></app-book>
```

```
book.component.ts <> book.component.html <> app.component.html <> index.html X
```

```
<> index.html > html
```

```
<meta charset="utf-8">
<title>ChulApp</title>
<base href="/">
<meta name="viewport" content="width=device-width, initial-scale=1
<link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
<app-root></app-root>
```

□ 3) 속성 바인딩

나. [속성명]="변수명"

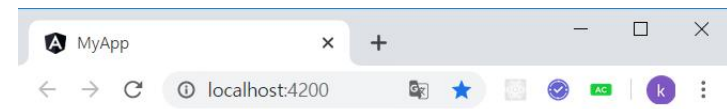
컴포넌트 클래스의 데이터를
template인 **HTML태그의 특정 속성값으로 전달할 때** 사용된다.

```
<h1>{{title}}</h1>
```

다음 이미지의 파일명과 width/height 값을 속성 바인딩으로 설정한다.

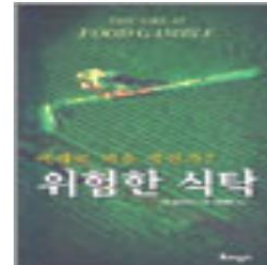

```
<img [src]="imgName" [width]="imgWidth" [height]="imgHeight">
```

```
export class AppComponent {
  title = '속성 바인딩 실습';
  imgName = "../assets/image/a.jpg";
  imgWidth = 200;
  imgHeight = 200;
}
```



속성 바인딩 실습

다음 이미지의 파일명과 width/height 값을 속성 바인딩으로 설정한다.



□ 4) class 바인딩

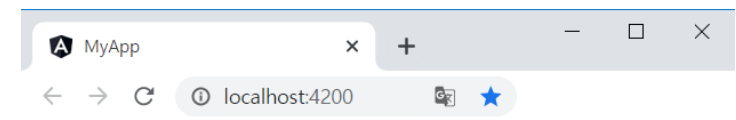
다. [class.CSS클래스명]=boolean값;

CSS 스타일을 적용할지 여부를 결정할 수 있다.
true인 경우에 CSS 스타일이 적용된다.

```
app.component.html ×
my-app > src > app > <> app.component.html > ...
1 <h1>{{title}}</h1>
2 <p [class.font-red]=true>홍길동</p>
3 <p [class.font-blue]=true>이순신</p>
4 <p [class.font-red]='result'>이순신</p>
5
```

```
export class AppComponent {
  title = 'CSS의 class 바인딩 실습';
  result = true;
}
```

```
# app.component.css ×
my-app > src > app > # app.component.
1 ul{
2   list-style: none;
3 }
4 .font-red{
5   color: red;
6 }
7 .font-blue{
8   color: blue;
9 }
```



CSS의 class 바인딩 실습

홍길동

이순신

이순신

❑ class바인딩



```
# app.component.css × <> app.comp
src > app > # app.component.css >
1  .font-red{
2    color: red;
3  }
4  .font-blue{
5    color: blue;
6  }
```

```
이동(G) 실행(R) 터미널(T) 도움말(H) app.component.ts - chu
TS app.component.ts × # app.component.css <> app.comp
src > app > TS app.component.ts > AppComponent
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css']
7  })
8  export class AppComponent {
9    title = 'chul-app';
10   result= false;
11 }
```

```
rc > app > <> app.component.html > p
1  <h1>class바인딩</h1>
2  <p [class.font-red]=true>[class.font-red]=true</p>
3  <p [class.font-red]=false>[class.font-red]=false</p>
4  <p [class.font-red]="result">app.component.ts의 result=false설정, [class.font-red]="result"</p>
```

새 창에서 열기 (Ctrl+클릭)

C:\angular_chul\chul-app>ng serve --open

□ 5) 이벤트 바인딩



라. (이벤트명)="callback()"

template에서 발생하는 다양한 형태의 이벤트를 처리하는 방법으로
컴포넌트 클래스의 메서드가 콜백으로 호출된다.

```
TS app.component.ts ×
my-app > src > app > TS app.component.ts > ...
5   templateUrl: './app.component.html',
6   styleUrls: ['./app.component.css']
7 })
8 export class AppComponent {
9   title = '이벤트 바인딩 실습';
10
11   handleEvent():void{
12     console.log("handleEvent")
13   }
14
15   handleEvent2(name:string):void{
16     console.log("handleEvent2", name)
17   }
18 }
```

```
<> app.component.html ×
my-app > src > app > <> app.component.html > ...
1   <h1>{{title}}</h1>
2   <button (click)="handleEvent()">
3     이벤트 처리
4   </button>
5   <button (click)="handleEvent2('홍길동')">
6     이벤트 처리2
7   </button>
```

□ Event바인딩(\$event- >event 객체의 전달)

```
app.component.html X TS app.component.ts
src > app > app.component.html > ...
1 <h1>{{title}}</h1>
2 <button (click)="handleEvent($event)">이벤트 처리 </button><br>
3 <!--(이벤트명)="메소드명()"-->
4 <button (click)="handleEvent2($event, '홍길동')>이벤트 처리2</button><!--매개변수-->
```

```
7
8 export class AppComponent {
9   title = '이벤트 바인딩 실습';
10  handleEvent(event):void{
11    console.log("handleEvent====", event);
12  }
13  handleEvent2(event, name:string ):void{
14    console.log("handleEvent2====", event, "\t", name );
15  }
16 }
17
```

□ 이벤트바인딩



```
//
export class AppComponent {
  title = 'chul-app';
  //이벤트 처리 메서드
  handleEvent():void{
    console.log("OK");
  }
  handleEvent2(name:string):void{
    console.log("OK2", name);
  }
  //optional parameter
  handleEvent3(name:string, age?:number):void{
    console.log("OK3", name, age);
  }
}
```

app > <> app.component.html > br

```
1 <h1>이벤트 처리 실습</h1>
2 <button (click)="handleEvent()">OK</button><br>
3 <button (Click)="handleEvent2('홍길동')>OK2-홍길동</button><br>
4 <button (click)="handleEvent3('홍길동', 20)">OK3-홍길동, 20</button><br> <!--
5 <button (click)="handleEvent3('홍길동')>0k3-홍길동</button><br>
```

The button element represents

[MDN Reference](#)

□ 5) 이벤트 바인딩

\$event 를 사용하여 이벤트 정보를 담고있는 이벤트 객체를 파라미터로 전달 할 수 있다.

```
app.component.ts X
- app > src > app > TS app.component.ts > ...
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css']
7  })
8  export class AppComponent {
9    title = '이벤트 바인딩 실습';
10
11    handleEvent(event):void{
12      console.log("handleEvent", event)
13    }
14    handleEvent2(event, name):void{
15      console.log("handleEvent2", event, name)
16    }
17  }
```

```
app.component.html X
- app > src > app > app.component.html > ...
1  <h1>{{title}}</h1>
2  <button (click)="handleEvent($event)">
3    |   이벤트 처리
4  </button>
5  <button (click)="handleEvent2($event, '홍길동')">
6    |   이벤트 처리
7  </button>
```

MyApp

localhost:4200

이벤트 바인딩 실습

이벤트 처리 이벤트 처리

Angular is running in the development mode.
[WDS] Live Reloading enabled.

handleEvent

```
MouseEvent {isTrusted: true, altKey: false, bubbles: true, button: 0, buttons: 0, cancelBubble: false, cancelable: true, clientX: 69, ...}
```

□ 단방향 바인딩 - 이벤트-체크박스, select

```
//
export class AppComponent {
  title = 'chul-app';

  //Call버튼에 대한 이벤트 처리
  callName(name:string):void{
    console.log(name);
  }
  handleEvent(name:string):void{
    console.log(name);
  }

  //checkbox 예제\
  flag= false;
  allCheck(f){ //true/false
    console.log("allCheck(f)=====", f);
    this.flag= f;
  }
}
```

```
<input #username type="text" placeholder="이름을입력하시오">
<button (click)="callName(username.value)">Call</button><br>
<h3> select</h3>
<select #fruit (change)="handleEvent(fruit.value)">
  <option>바나나</option>
  <option>사과</option>
  <option>옥수수</option>
  <option>감자</option>
</select><br>
<h3>checkbox실습</h3>
전체:<input type="checkbox" #all (click)="allCheck(all.checked)"><br>
야구:<input type="checkbox" [checked]=all.checked><br>
농구:<input type="checkbox" [checked]=flag><br>
축구:<input type="checkbox" [checked]=all.checked><br>
```

❑ 6) 양방향 바인딩

마. [(ngModel)]=“변수명”;

- template과 컴포넌트 클래스간에 양방향(two-way) 바인딩 처리 가능.
template은 사용자 입력을 처리할 수 있는 위젯을 사용한다.

- 반드시 **FormsModule**이 필요하다.(app.module.ts)

```
TS app.module.ts X
my-app > src > app > TS app.module.ts > ...
1  import { BrowserModule } from '@angular/platform-browser';
2  import { NgModule } from '@angular/core';
3  import { FormsModule } from '@angular/forms';
4  import { AppComponent } from './app.component';
5
6  @NgModule({
7    declarations: [
8      AppComponent
9    ],
10   imports: [
11     BrowserModule,
12     FormsModule
13   ],
```



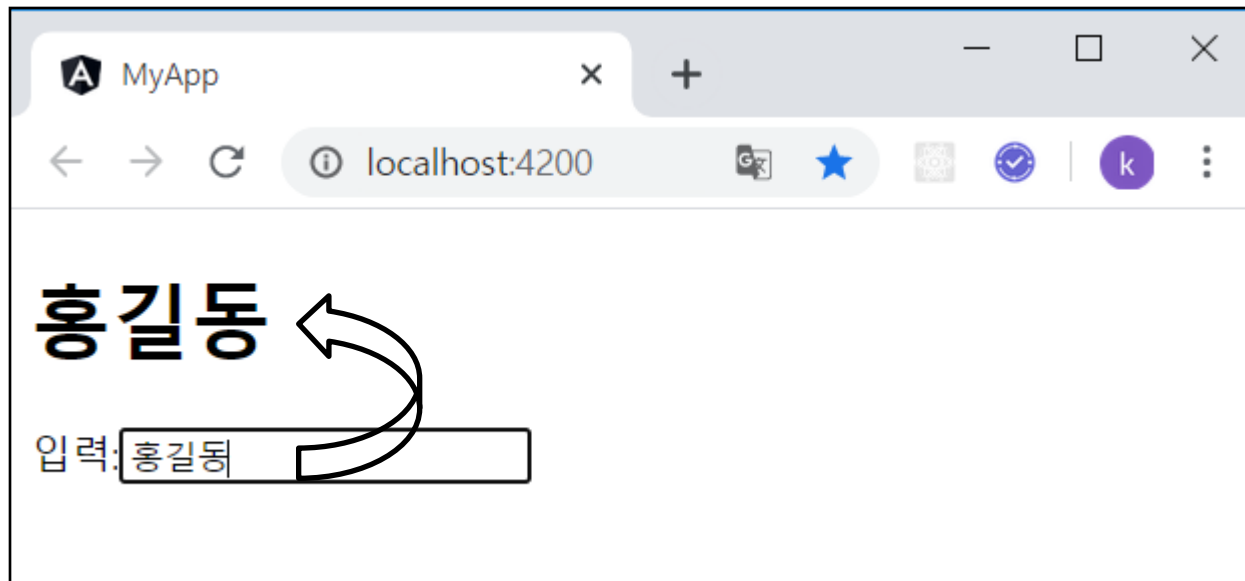
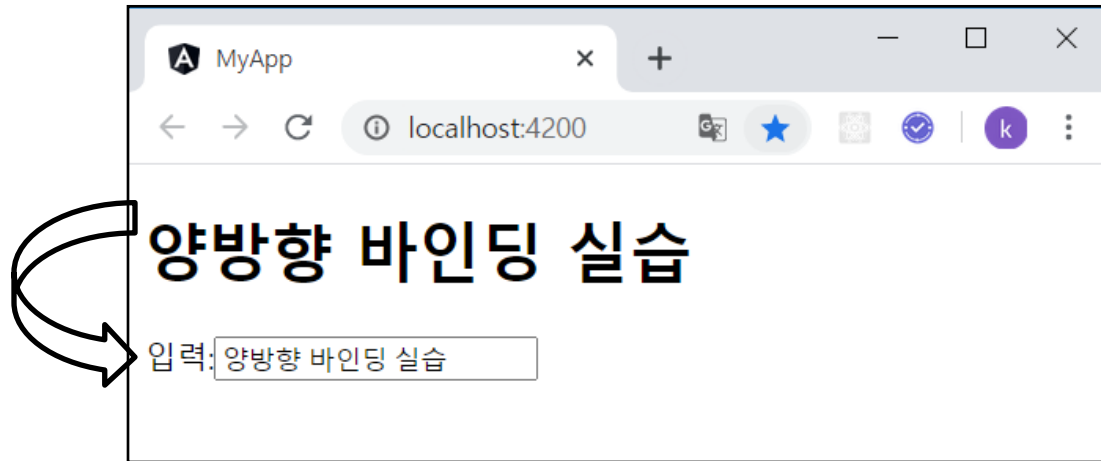
src > app > src/app.module.ts > app.module.ts

```
1 import { BrowserModule } from '@angular/platform-browser';
2 import { NgModule } from '@angular/core';
3 import { AppRoutingModule } from './app-routing.module';
4 import { AppComponent } from './app.component';
5 //FormsModule import
6 import {FormsModule} from "@angular/forms";
7
8 @NgModule({
9   declarations: [
10     AppComponent
11   ],
12   imports: [
13     BrowserModule,
14     AppRoutingModule,
15     FormsModule //FormsModule 사용
16   ],
17   providers: [],
18   bootstrap: [AppComponent]
19 })
20 export class AppModule { }
```

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

```
1 <h1>양방향 바인딩</h1>
2 입력:<input type="text" [(ngModel)]="title"><br><!--component.title변수-->
3 출력값:{{title}}
```

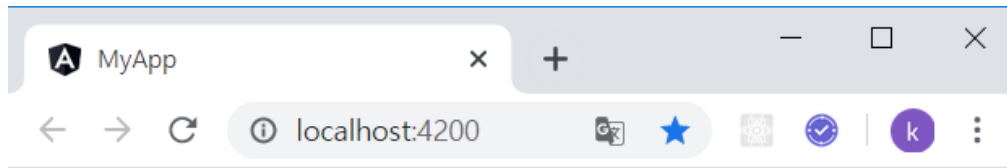

❑ 6) 양방향 바인딩



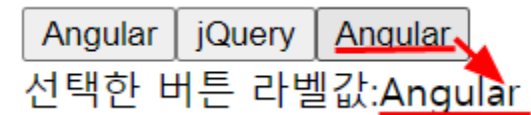
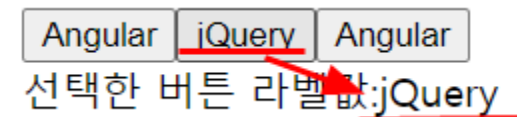
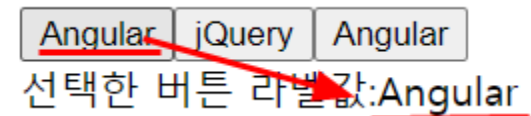
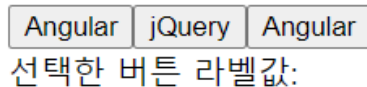
□ 실습 문제 1

다음과 같은 이벤트 처리를 위한 어플리케이션을 작성하시오.

- 각 버튼을 클릭하면 해당 라벨값을 출력한다.
- 이벤트 처리를 위한 콜백 메서드는 반드시 하나로 처리한다.



이벤트 바인딩 실습예제 1



```

src > app > < app.component.html > ...
1 <h1>{{title}}</h1>
2 <!--app.moudles.ts 에 FormsModule이 필요함-->
3 <button (click)="handleEvent($event)">Angular</button>
4 <button (click)="handleEvent($event)">jQuery</button>
5 <button (click)="handleEvent($event)">Vu</button>
6 <br>
7 선택한 버튼 라벨값 : {{btnLabel}}
  
```

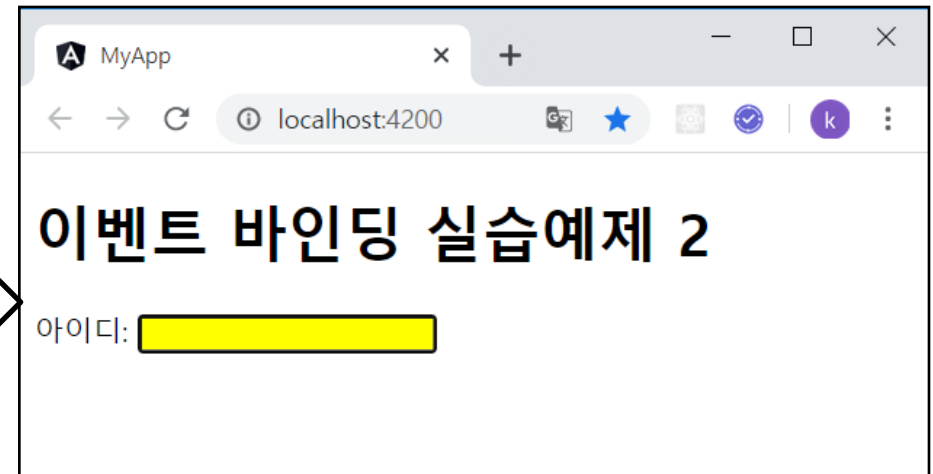
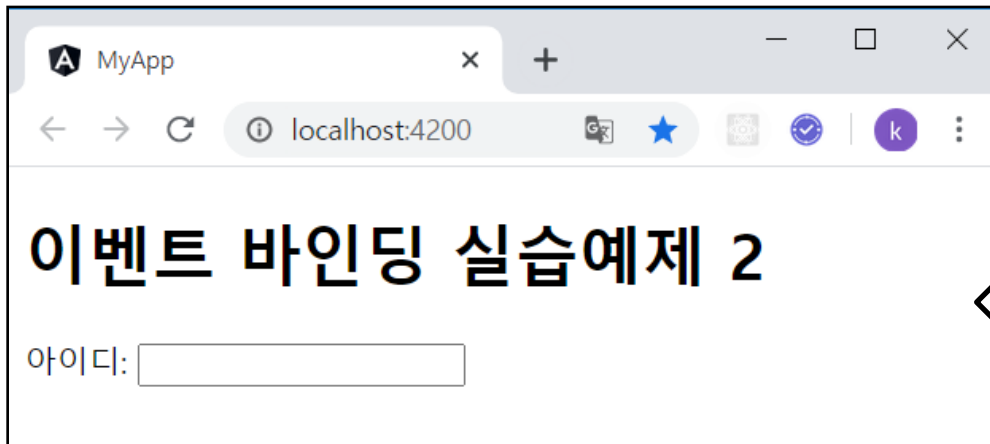
```

2
3 @Component({
4   selector: 'app-root',
5   templateUrl: './app.component.html',
6   styleUrls: ['./app.component.css']
7 })
8 export class AppComponent {
9   title="이벤트 바인딩 실습예제1";
10  btnLabel="";
11  handleEvent(event){
12    this.btnLabel=event.target.innerText;
13  }
14
  
```

□ 실습 문제 2



다음과 같은 초기 화면에서 input 태그가 focus를 받았을 때 배경색을 yellow로 설정하고 focus를 잃었을 때(blur) 배경색을 제거하는 어플리케이션을 구현 하시오.





```
src > app > # app.component.css > .input_color
1  .input_color{
2      background-color: yellow;
3  }
```

```
src > app > <> app.component.html > input
1  <h1>이벤트 바인딩 실습문제2, class 바인딩</h1>
2  아이디 : <input type="text" (focus)="handleEvent(true)"
3           (blur)="handleEvent(false)"
4           [class.input_color]="result" />
```

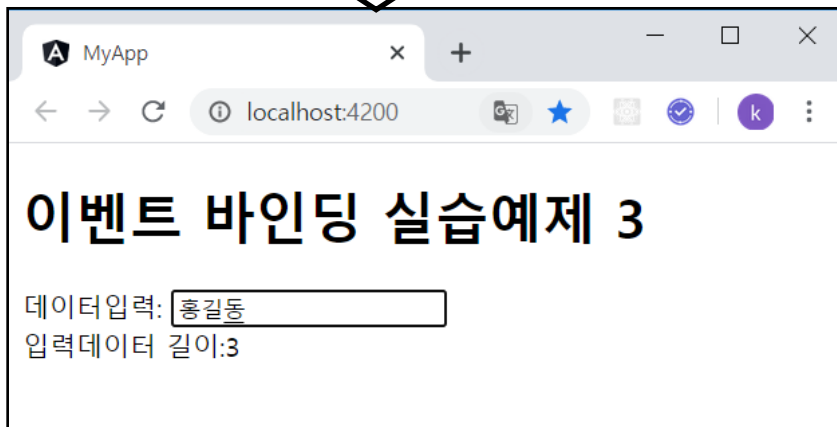
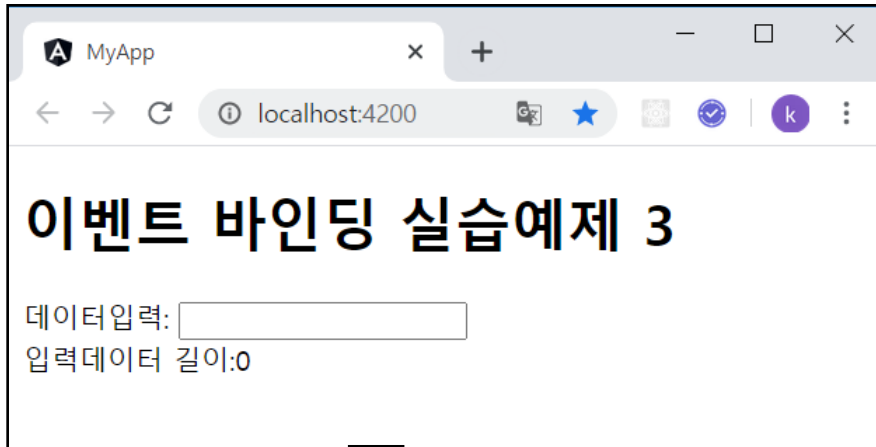
ng serve --port=4201 --open

ngular_chul\chul-app>ng serve --port=4201 --open

□ 실습 문제 3



다음과 같은 초기 화면에서 input 태그에 값을 입력할 때,
입력한 문자열의 길이를 출력하는 코드를 구현 하시오.







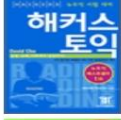

□ 실습 문제 4

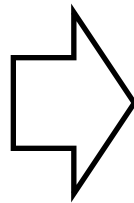
다음과 같이 검색란에 임의의 값을 입력시, 한 글자라도 일치하는 목록을 반환하는 어플리케이션을 구현 하시오.

- 검색값을 입력하지 않은 경우에는 모든 도서 목록을 보여준다.

도서 목록 6권

검색:

-  위험한 식탁
-  공부의 비결
-  오메르타
-  행복한 여행
-  해커스 토익
-  도로 안내서



도서 목록 6권

검색:

-  위험한 식탁
-  행복한 여행

```
C:\angular_chul\chul-app>ng g component book
CREATE src/app/book/book.component.html (19 bytes)
CREATE src/app/book/book.component.spec.ts (612 bytes)
CREATE src/app/book/book.component.ts (267 bytes)
CREATE src/app/book/book.component.css (0 bytes)
UPDATE src/app/app.module.ts (167 bytes)
```

```
app / src/app.component.html / ui / ii
<h1>{{titleName}}{{books.length}}권</h1>
검색: <input type="text" #xyz (keyup)="handleEvent(xyz.value, $event)"/>
<h2>클릭한 도서목록</h2>
도서id: {{item.id}}<br>
도서명: {{item.name}}<br>
도서가격: {{item.price}}<br>
<ul>
  <li *ngFor="let book of bookResult">
    <a (click)="imgClick(book.id)">
      
    </a>
    {{book.name}}
  </li>
</ul>
```




```
export class AppComponent {
  title = 'chul-app';
  titleName:string="도서 목록";
  bookResult=[];
  constructor(){
    for(var book of this.books){
      this.bookResult.push(book); //초기화
    }
  }
  books=[
    {id:"p01", name:"위험한 식탁", price:2000, date:"20170202", img:"a.jpg"},
    {id:"p02", name:"공부의 비결", price:3000, date:"20170203", img:"b.jpg"},
    {id:"p03", name:"오메르타", price:4000, date:"20170204", img:"c.jpg"},
    {id:"p04", name:"행복한 여행", price:5000, date:"20170205", img:"d.jpg"},
    {id:"p05", name:"해커스 토익", price:6000, date:"20170206", img:"e.jpg"},
    {id:"p06", name:"도로안내서", price:7000, date:"20170207", img:"f.jpg"},
  ];
}
```



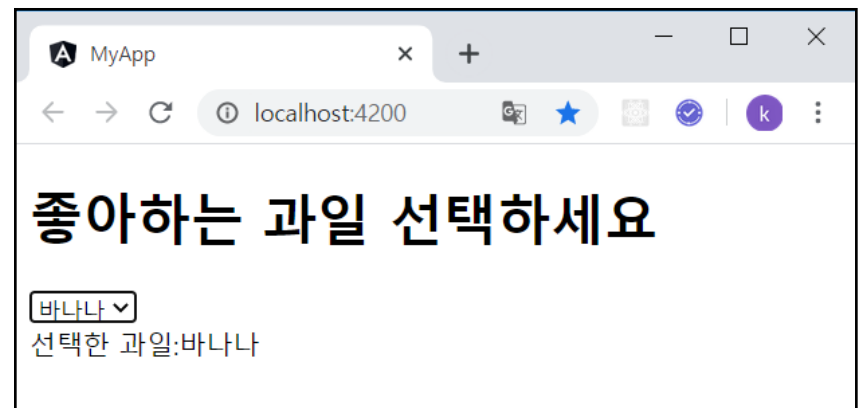
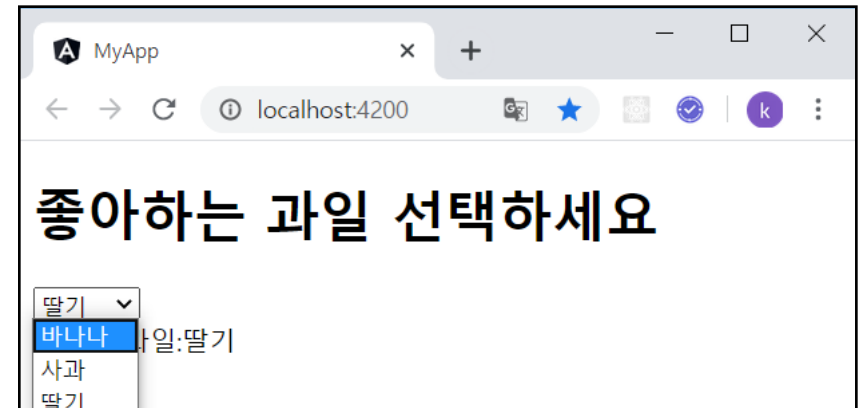
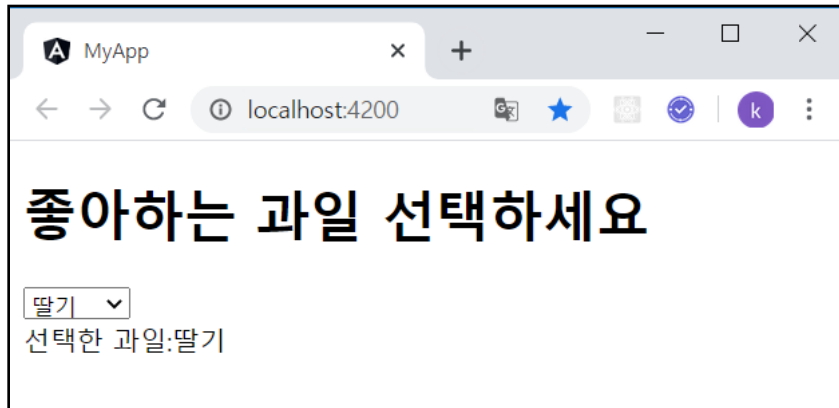
```
},
//검색관련이벤트 메서드
handleEvent(inputData, $event){
    var searchName= inputData;
    //var searchName= $event.target.value;
    this.bookResult=[]; //초기화 작업
    if(searchName==''){
        for(var book of this.books){
            this.bookResult.push(book);
        }
    }else{
        for(var book of this.books){
            if(book.name.indexOf(searchName) != -1){
                this.bookResult.push(book);
            }
        }
    }//end if
}//end HandleEvent
```

```
item={
    id:"",
    name:"",
    price:0,
    date:"",
    img:""
}
//item={}
imgClick(id){
    for(var book of this.bookResult){
        if(book.id==id){
            this.item= book;
        }
    }
}
```

□ 실습 문제 5



다음과 같이 select 태그를 선택할 때, 선택된 값이 출력되도록 어플리케이션을 구현하시오.
(단, 초기 선택값은 ‘딸기’)





❑ 실습5-*app.modules.ts*에 *FormModule* 추가

```
TS app.module.ts X TS app.component.ts <> app.component.html
src_0_template > app > TS app.module.ts > AppModule
1  import { BrowserModule } from '@angular/platform-browser';
2  import { NgModule } from '@angular/core';
3
4  import { AppRoutingModule } from './app-routing.module';
5  import { AppComponent } from './app.component';
6  import { FormsModule } from '@angular/forms'
7
8  @NgModule({
9    declarations: [
10     AppComponent
11   ],
12   imports: [
13     BrowserModule,
14     AppRoutingModule,
15     FormsModule
16   ],
17   providers: [],
18   bootstrap: [AppComponent]
```

```
<> app.component.html X
src > app > <> app.component.html > ...
1  <h1>{{title}}</h1>
2  <select [(ngModel)]="fruit" >
3    <option>바나나</option>
4    <option>사과</option>
5    <option>딸기</option>
6    <option>수박</option>
7  </select>
8
9  선택한 과일 : {{fruit}}
```



수고하셨습니다.
