Template & Bindig

Template & TemplateUrl

```
selector: 'a01Component',
template: `
    <div class="panel">
            …
    </div>
</div>
`
```

OR

```
selector: 'a01Component',
templateUrl: 'HTML 파일 경로‘
```

* 파일 경로는 Component를 사용하는 HTML 파일 기준으로 설정한다.

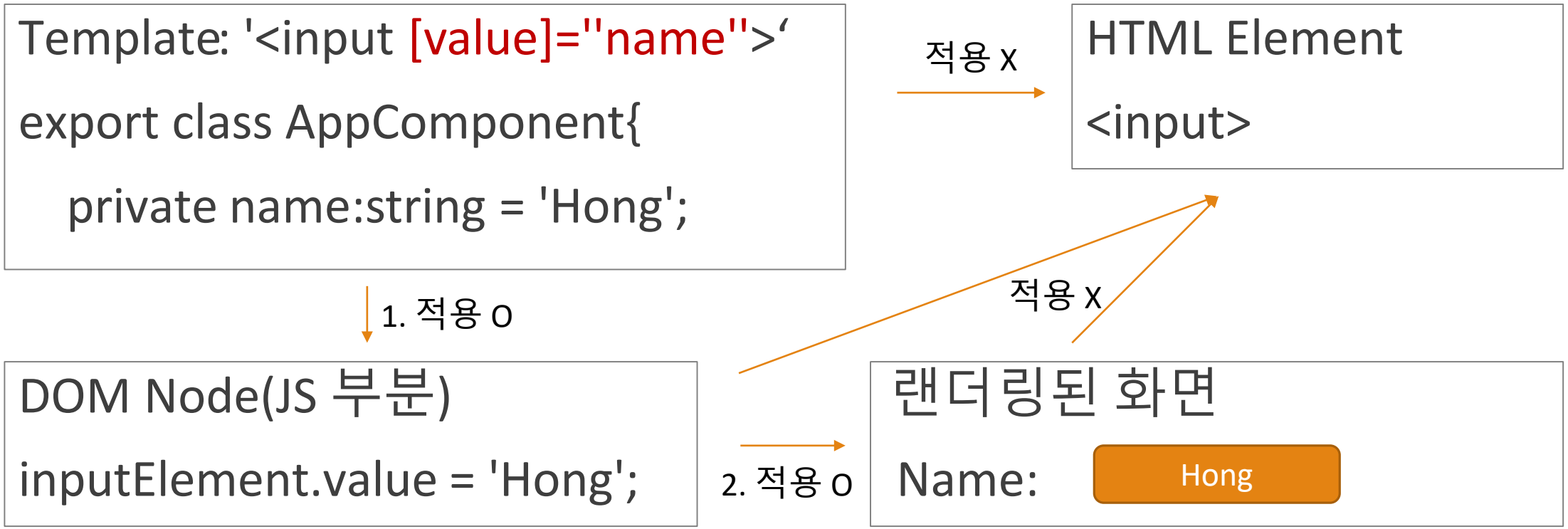
Binding

Binding 방식:

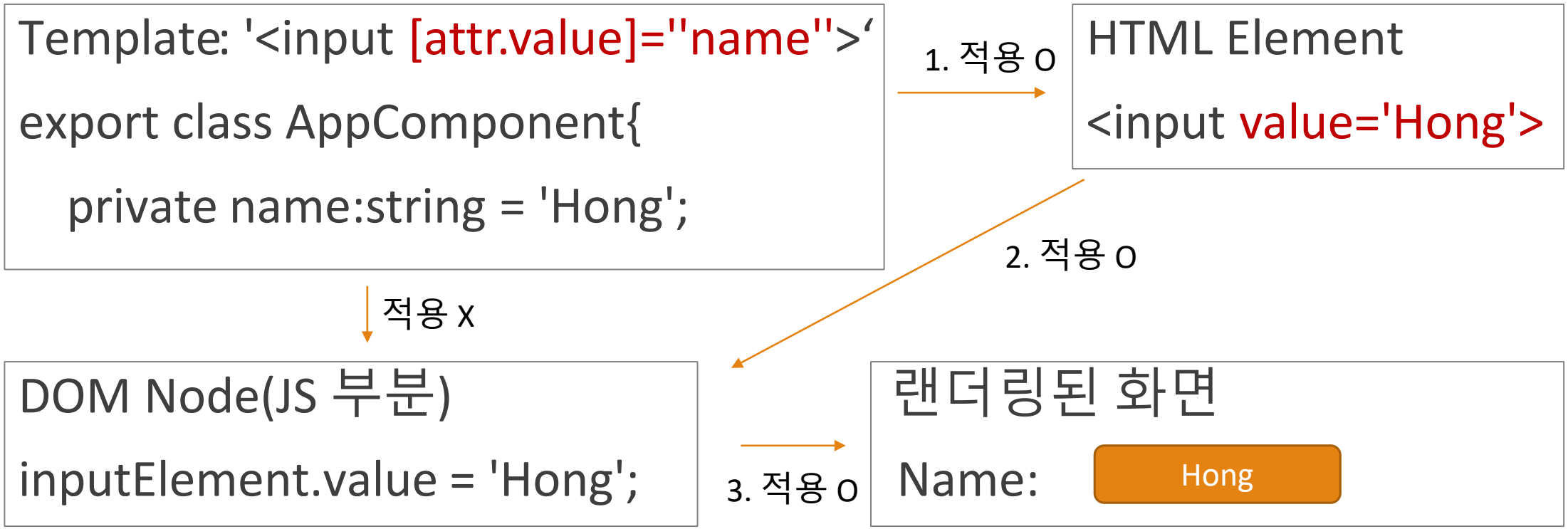
<tab>{{ 변수 명}}</tag>

- 바인딩 내부에서 연산자 사용은 가능하다
- AngularJS와 같이 바인딩 {{ }} 안에서 변수 선언을 할 수 없다.
- 지정한 변수 명이 없는 경우 Error 표시 없이 아무런 값도 표시하지 않는다.
- Object 타입의 경우 ?(없을 수 있음)으로 미 선언된 객체에 대한 에러 방지

속성 Binding


Template: '<input [value]="name">‘
export class AppComponent{
private name:string = 'Hong';
적용 X
HTML Element
<input>
1. 적용 O
적용 X
DOM Node(JS 부분)
inputElement.value = 'Hong';
2. 적용 O
랜더링된 화면
Name:
Hong

속성 Binding


Template: '<input [attr.value]=''name''>‘
export class AppComponent{
private name:string = 'Hong';
1. 적용 O
HTML Element
<input value='Hong'>
2. 적용 O
적용 X
DOM Node(JS 부분)
inputElement.value = 'Hong';
3. 적용 O
랜더링된 화면
Name:
Hong

ngModel

보통 Form Element에 사용.

```
<input class="form-control" [(ngModel)]="name">
```

- 감시자를 등록하므로 리소스 소모가 많다
- [value]="name" 과 (input)="Method($event) "가 결합된 형태.

ngIf / ngSwitch

*ngIf

조건에 맞는 요소를 DOM에서 삭제 또는 생성.

ngSwitch

조건에 맞는 요소를 CSS를 이용하여 표시 또는 비 표시/