1 way binding :

(class) .ts (template) .html

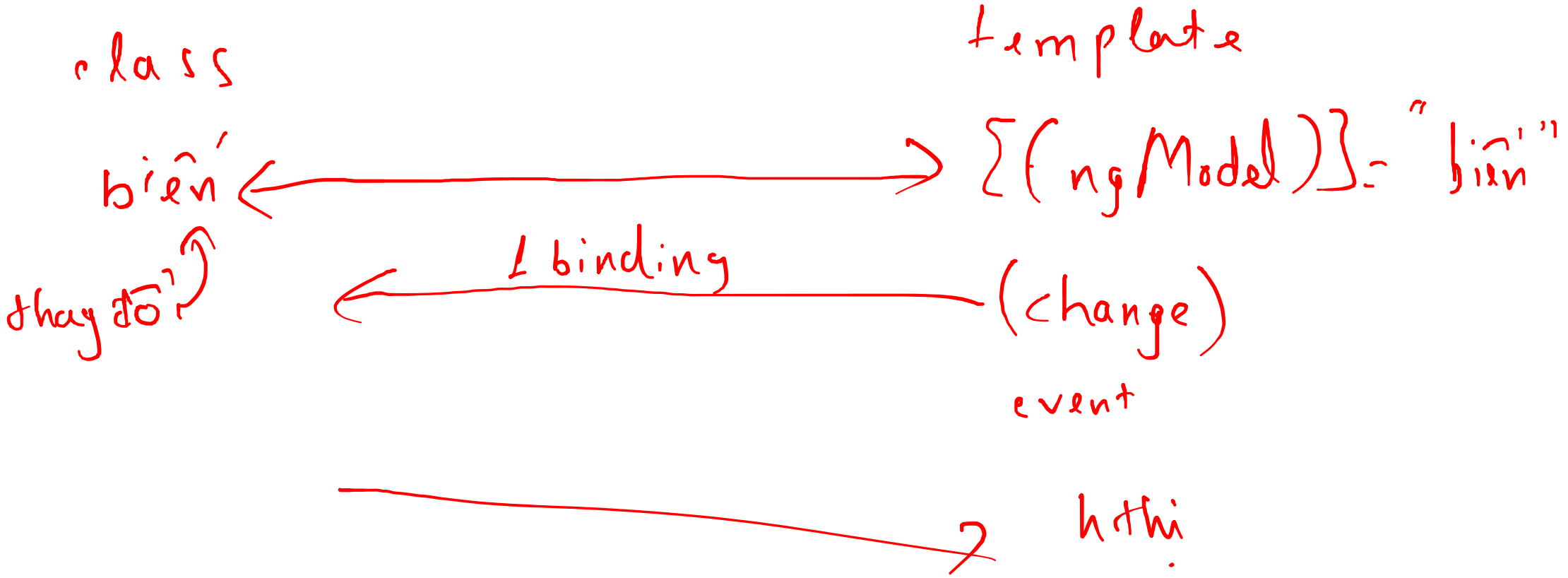
biến --1 way binding--> {{ biến }}

[src] = "biến"

truyền dữ liệu liên tục (giá trị của biến thay đổi)

⇒ data ngoài template thay đổi ngay lập tức

2 way binding



app.component.html M | post-unit.component.html U

src > app > app.component.html > div > app-post-unit

```
1  <div *ngFor="let item of posts">
2  <app-post-unit [postData]="item"></app-post-unit>
3  </div>
4
```

① ② ③

app.component.html M | post-unit.component.html U | post-unit.component.ts U

src > app > components > post-unit > post-unit.component.ts > PostUnitComponent > constru

```
1  import { Component, OnInit, Input } from '@angular/core';
2
3  @Component({
4    selector: 'app-post-unit',
5    templateUrl: './post-unit.component.html',
6    styleUrls: ['./post-unit.component.css']
7  })
8  export class PostUnitComponent implements OnInit {
9    @Input() postData: any;
10   constructor() { }
11
```

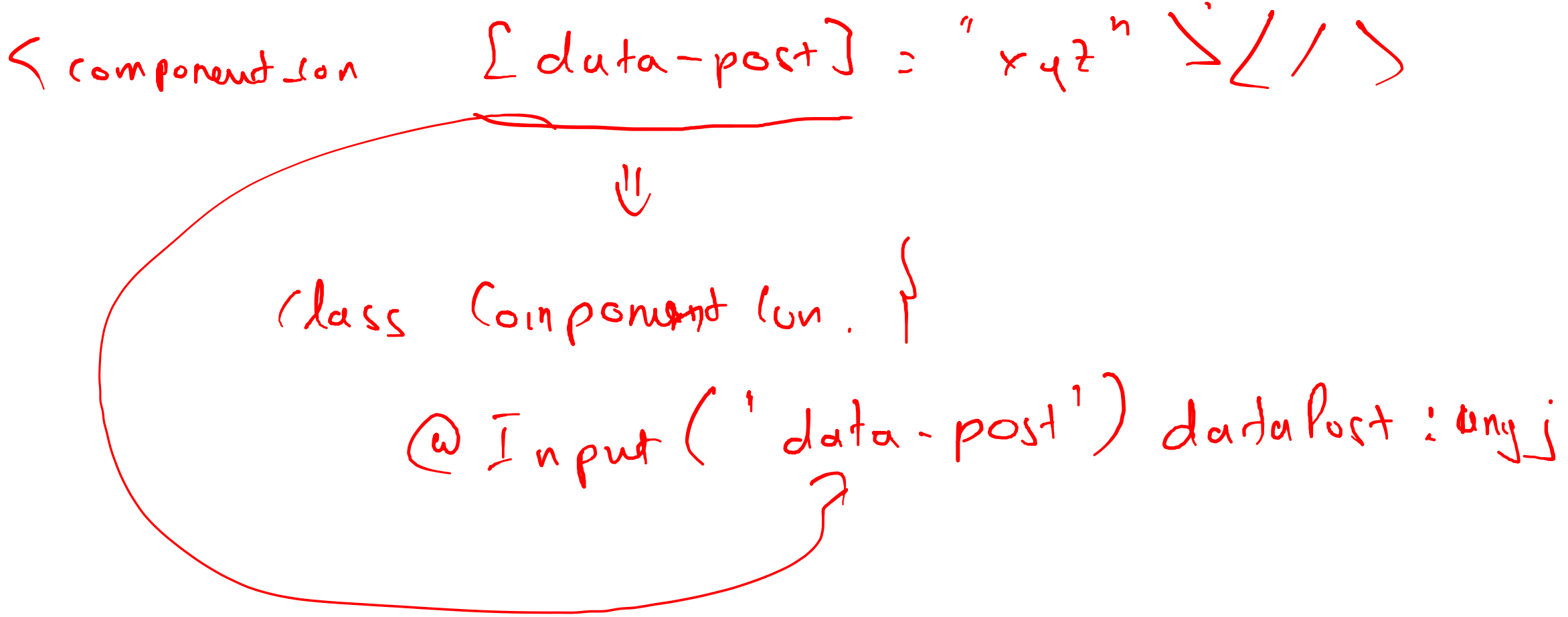
② : tạo ra thuộc tính mới.

Hứng data từ lớp cha

app.component.html M | post-unit.component.html U

src > app > components > post-unit > post-unit.component.html

```
1  <p>{{postData.title}}</p>
2
```


<component-ion [data-post] = "xyz" ></component-ion>
class ComponentIon {
@Input('data-post') dataPost : any;

cha.ts

xyz() { }

(5)

cha.html

<con (onRemove) = "xyz()"></con>

(3)

con.ts

@Output onRemove

abc() {

this.onRemove.emit()

(2)

con.html.

<button (click) = "abc()"> p

(1)