**@Input**

**@Output**

1. 父元件和1~多個子元件間的共享資料,可用

* Input:把某Class欄位標示為輸入屬性,並提供設定metadata(元資料)，屬性繫結到樣板的某個DOM屬性,當變更DOM屬性值,自動將此值用來更新輸入屬性 (DOM變動->變動)
* Output: 把某Class欄位標示為輸出屬性,並提供設定metadata，輸出屬性的值發生改變時,凡繫結到輸出屬性的DOM屬性會被自動更新

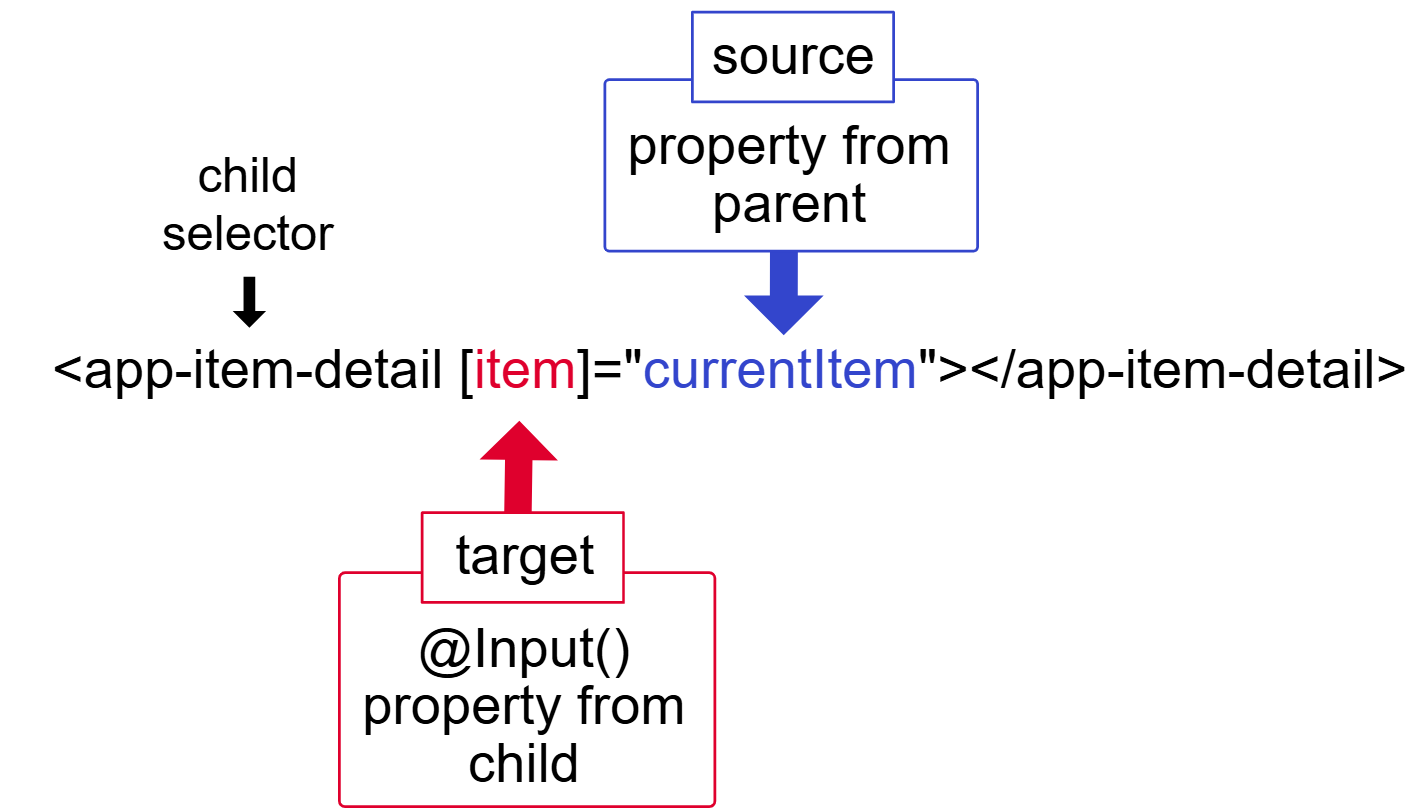
(Class屬性變動->DOM變動)

1. 資料傳送方式設定

* input: 父元件傳給子元件:

1. 先設定小孩,在class中宣告@Input()的屬性
2. 在爸爸的html設定小孩的selector 並在標籤中設定[小孩的屬性]="爸爸的屬性"
3. 在爸爸的class設定屬性和值

此會單向給資料到小孩設定的@Input()



* output: 子元件傳給父元件: (output的資料需要的type: [EventEmitter](https://angular.tw/api/core/EventEmitter))

範例以子元件中輸入資料,把資料傳遞給父元件

1. 先import { Output, EventEmitter } from '@angular/core';
2. 子元件class中增加物件

@Output() forParentData = new EventEmitter<string>();

1. 子元件class增加一個方法,把資料從input標籤放入傳遞的屬性中

addMsgToParent(msg: string){

this.forParentData.emit(msg);

}

1. 子元件的html增加input

<label>

<span>Add msg to parent: </span><br/>

<input type="text" #newMsg>

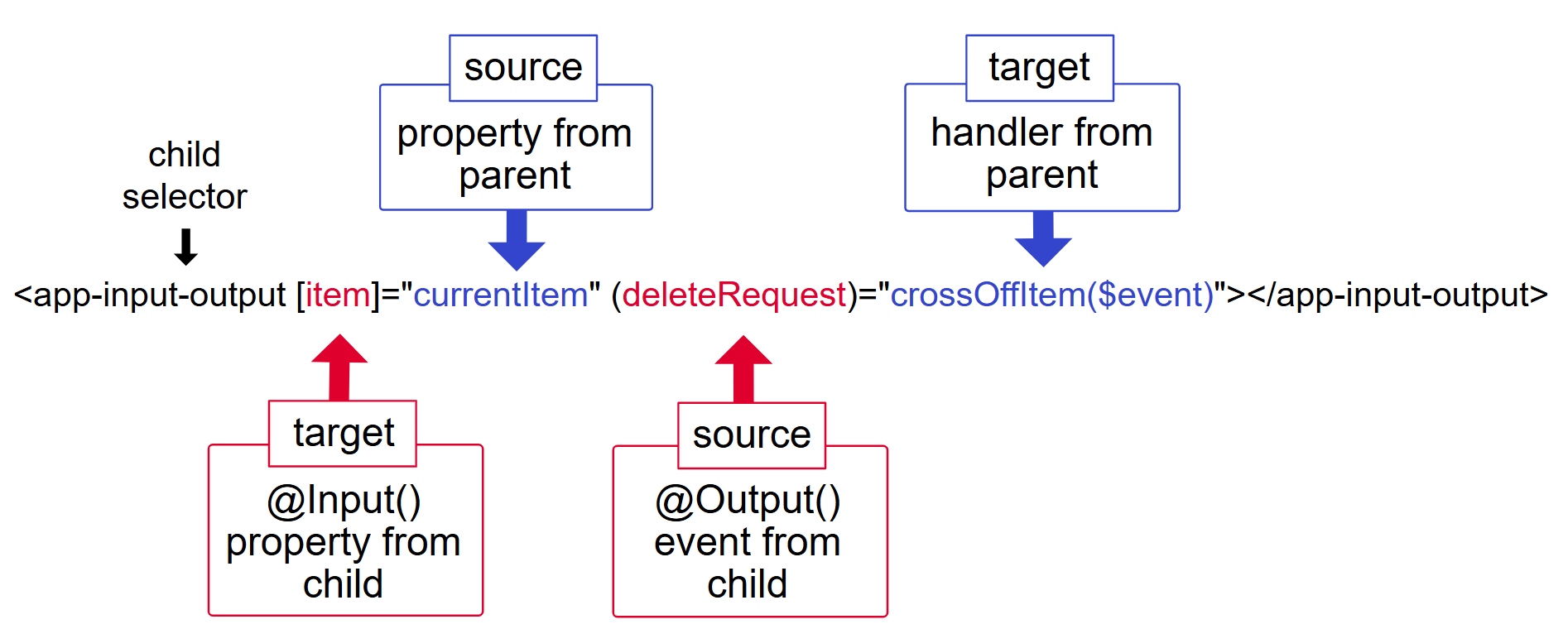
<button type="button" (click)="addMsgToParent(newMsg.value)">add msg</button>

</label>

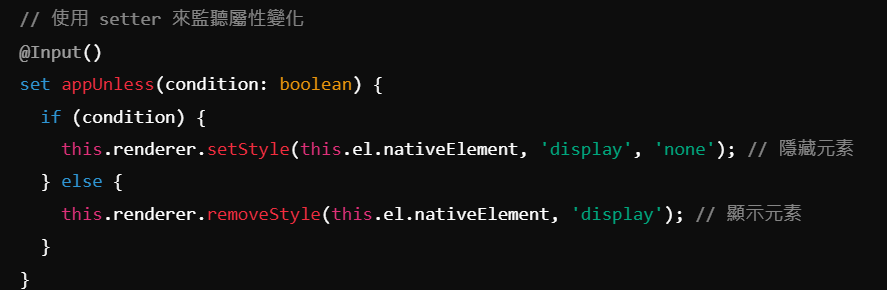
1. 父元件中增加一個屬性和一個放資料到此屬性的方法
2. 父元件的html中設定子元件的output為事件,並設定父元件的方法

<child-selector (forParentData)="addMsg($event)"></child-selector>

可以同時設定@Input和@Output在同個html標籤上



1. input可以設定set 來取代方法ngOnChanges 監聽變化



**@ViewChild**

建立動態樣板使用

<ng-template #myTemplate>

<p>This is a dynamic template content!</p>

</ng-template>

export class DirectivesComponent {

@ViewChild('myTemplate', {static : true}) dynamicTemplate:TemplateRef<any> | null = null;

**@ HostListener**

<https://angular.tw/api/core/HostListener>

建立監聽事件

@HostListener('click')

onClickListener(){

      this.hightLight('red');

}

**@Directive**

* 自訂義指令(元件,結構型指令,屬性型指令)
  + 元件:建立可重複用的UI元件

Component為Directive擴展

ex: <app-my-component></ app-my-component >

* + 結構型指令:新增或刪除DOM節點

ex: <div \*appMyIf="ture">…</div>

* + 屬性型指令:改變DOM元素的外觀或行為

ex:<div [appHeightLight]="red">…</div>

* 產生此下的CLI指令:ng g d 名稱
* 建立屬性型指令:

1.設定css

selector:設定[xxx]

@Directive({

  selector: '[appMyDirective]'

})

class中建立內容

範例為元素的背景改為黃色

export class MyDirectiveDirective {

  constructor(e : ElementRef) {

    e.nativeElement.style.backgroundColor = 'yellow';

   }

}

<div appMyDirective>MyDirective: hightLight</div>



2.建立監聽事件的屬性指令

此範例為click元素背景變成紅色,一秒後變回黃色

用@HostListener()建立監聽事件

@Directive({

  selector: '[appMyDirective]'

})

export class MyDirectiveDirective {

  constructor(private e : ElementRef) {

    this.e.nativeElement.style.backgroundColor = 'yellow';

   }

@HostListener('click')

onClickListener(){

      this.hightLight('red');

}

//e在建構子那邊要設定private之後才能用,因為變成類的property

hightLight(backgroundColorString : string) {

    this.e.nativeElement.style.backgroundColor = backgroundColorString;

    setTimeout(()=>{

    this.e.nativeElement.style.backgroundColor = 'yellow';

    }, 1000)

}

3.建構子完成後才會設定屬性

可以使用@input setter替換掉ngOnChanges

<div appMyDirective hightLightColor="blue">MyDirective: hightLight Blue</div>

import { Directive, ElementRef, HostListener, Input , OnChanges, SimpleChanges} from '@angular/core';

@Directive({

  selector: '[appMyDirective]'

})

export class MyDirectiveDirective {

  constructor(private e : ElementRef) {

    this.e.nativeElement.style.backgroundColor = 'yellow';

   }

   @HostListener('click') onClickListener(){

      this.hightLight('red');

   }

   //e在建構子那邊要設定private之後才能用,因為變成類的property

   hightLight(backgroundColorString : string) {

    this.e.nativeElement.style.backgroundColor = backgroundColorString;

    setTimeout(()=>{

      this.e.nativeElement.style.backgroundColor = 'yellow';

    }, 1000)

   }

   @Input() hightLightColor !:string;

  ngOnChanges(change: SimpleChanges){

    if(change['hightLightColor']){

      this.e.nativeElement.style.backgroundColor = this.hightLightColor;

    }

  }

}

