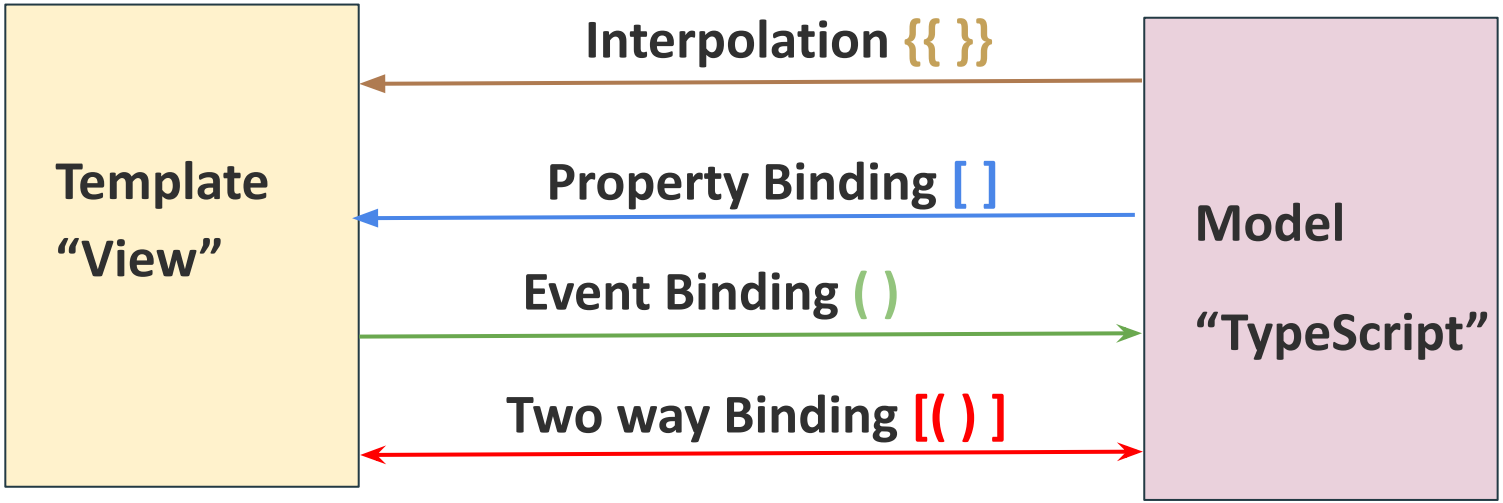
Data binding

One way binding & Two way binding

C quoi Data binding ?

Le Data Binding est un élément essentiel dans les frameworks SPA (Single Page Application), il permet la synchronisation entre la vue et le modèle TypeScript ,voici un schéma général du fonctionnement du Data Binding.



Plan : “One way binding “

1- String interpolation
2- Property binding
3- event binding

String interpolation : “typeScript vers HTML”

- C'est un type de databinding il fait partie de la catégorie **Unidirectionnel (Du Modèle vers la template html).**

- Donc on peut l’utiliser pour évaluer la valeur d’un membre **(Propriété ou méthode)** du component. Il suffit, tout simplement, de mettre le nom de la propriété du component dans double accolades ouvrantes et fermante comme suivant {{ }}, et angular va se charger de chercher dans le component puis l'évaluer dans l’affichage.

Exemple “String interpolation “

- Et on va utiliser l’interpolation pour envoyer des données à partir du component vers la template graphique ‘html’

commande : `ng g c components/interpolation --skipTests=true --inlineStyle=true`

Property binding : “typeScript vers HTML”

Le "property binding" permet d'utiliser une propriété créée également côté TypeScript, pour en faire la valeur d'un attribut d'un élément du DOM (cas de communication de Component vers DOM du template également) via des crochet

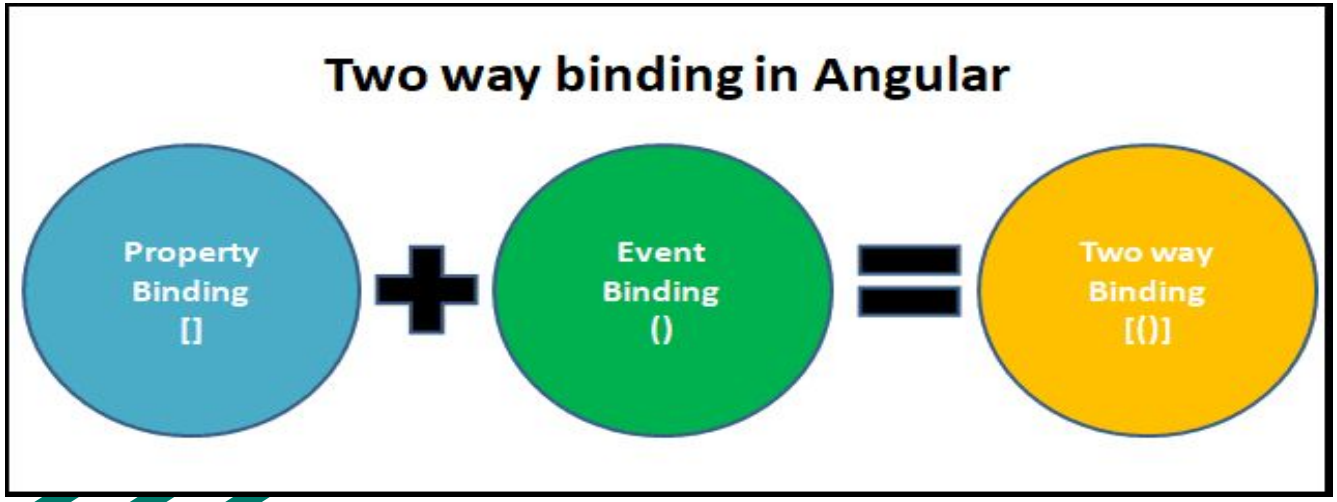
event binding : “HTML vers typeScript ”

- À partir d’un évènement du DOM, on peut interagir avec un composant,

  pour réaliser ce Data Binding, Angular utilise les évènements, d’où le nom d’Event Binding, avec ce mécanisme, vous pouvez être averti des évènements utilisateurs tels que le click, le frappe sur le clavier.. etc

Two Way Binding

Ce mécanisme permet de modifier le modèle à partir du DOM et de modifier le DOM à partir du modèle. “ liaison bidirectionnel “



Plan : “Two way binding “

1- @ViewChild
2- @ContentChild
3- @Input()
4- @Output() / @EventEmetteur<>()

@ViewChild :

ViewChild permet d'accéder **à** un composant enfant et appeler des méthodes ou d'accéder **à** des variables d'instance qui sont disponibles pour l'enfant

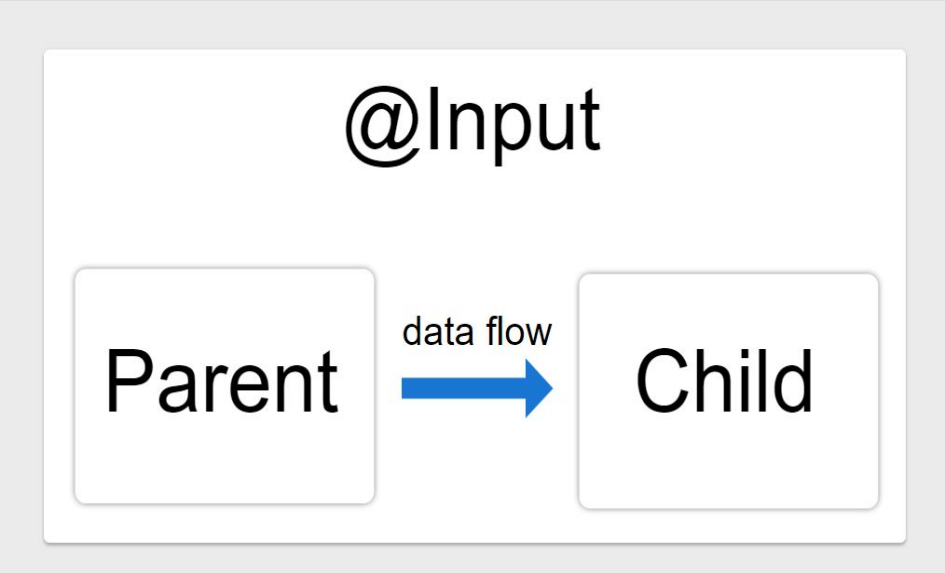
@ContentChild :

La projection de contenu est un motif dans lequel vous insérez, ou *projetez* , le contenu que vous souhaitez utiliser à l'intérieur d'un autre composant.

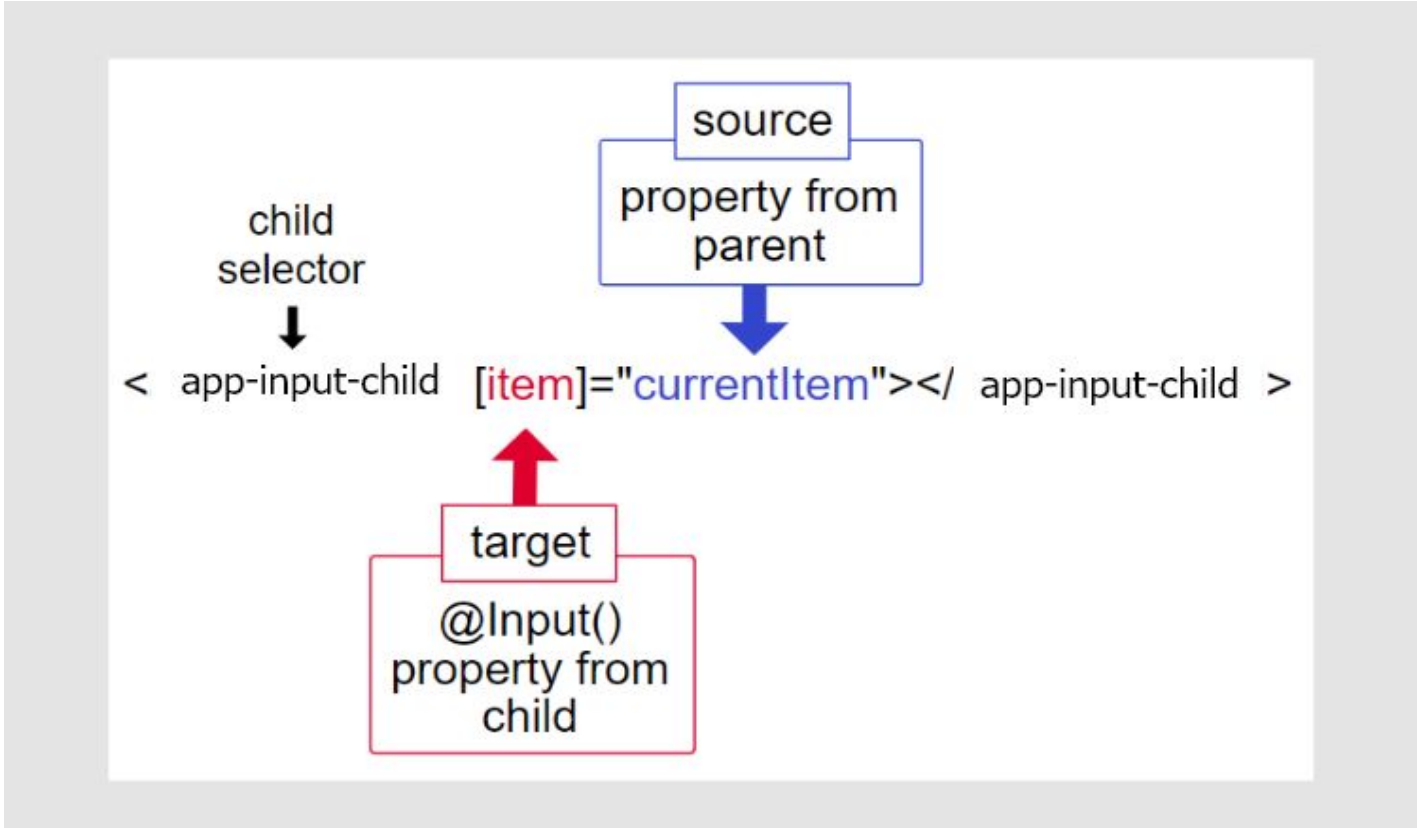
Par exemple, vous pouvez avoir un composant Card qui accepte le contenu fourni par un autre composant.

@input ()

Le décorateur dans un composant enfant ou une directive signifie que la propriété peut recevoir sa valeur de son composant parent.@Input() —-> **Angular transmet la valeur de la propriété à l'enfant**



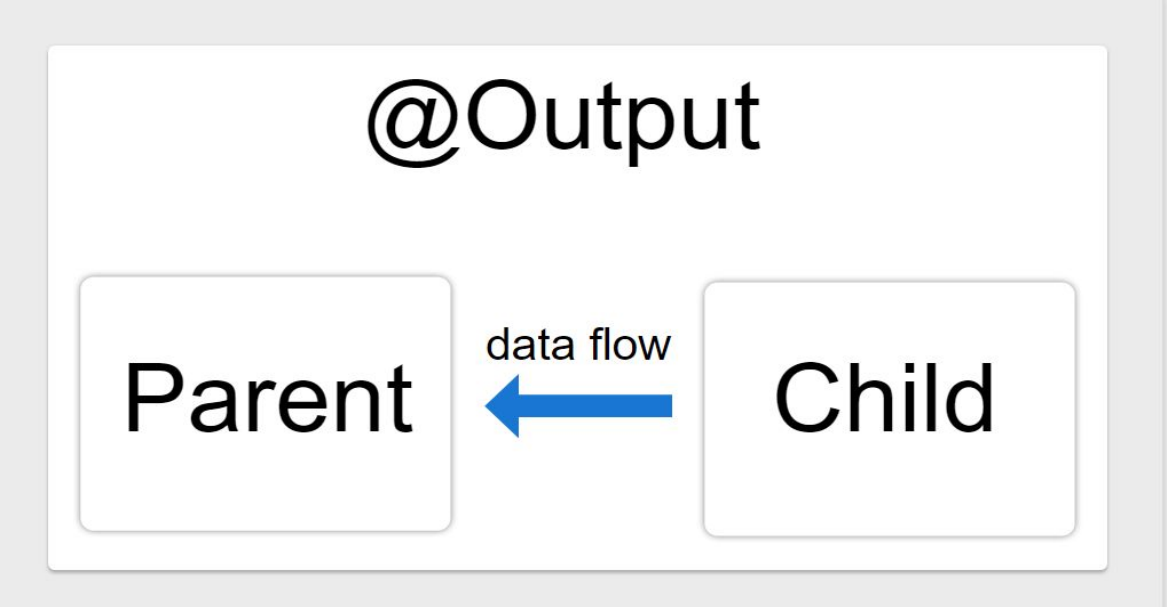
@input ()


source
property from parent
child selector
< app-input-child [item]="currentItem"></ app-input-child >
target
@Input() property from child

@Output()

Le décorateur d'un composant ou d'une directive enfant permet aux données de circuler de l'enfant vers le parent.@Output()

—-> **Angular transmet la valeur de l'enfant vers le parent**



@EventEmitter<>()

EventEmitter : est un module qui permet de partager des données entre les composants à l'aide des méthodes emit() et subscribe()
EventEmitter se trouve dans la couche Observables, qui observe les changements et les valeurs et émet les données vers les composants abonnés à cette instance EventEmitter.

—---> **emit() :** est une méthode EventEmitter qui émet un événement contenant une valeur donnée. emit() :n'a qu'un seul paramètre, value.

```
# angular
emit(value?: A): void
```

NgModel

- **NgModel** est une directive intégrée qui crée une instance FormControl à partir du modèle de domaine et la lie à un élément de contrôle de formulaire.
- NgModel lié la valeur des contrôles HTML (entrée, sélection, zone de texte) aux données d'application.
- L'Angular utilise la directive ngModel pour réaliser la liaison bidirectionnelle sur les éléments du formulaire HTML. Il se lie à un élément de forme analogue **input**, **select**, **selectarea**. etc.

[(ngModel)] effectue une liaison bidirectionnelle pour la lecture et l'écriture des valeurs de contrôle d'entrée. Si une directive [(ngModel)] est utilisée, le champ d'entrée prend une valeur initiale dans la classe de composant liée et la remet à jour chaque fois qu'une modification de la valeur du contrôle d'entrée est détectée (au clavier et par pression du bouton).

MERCI POUR VOTRE ATTENTION